

CS6701-

CRYPTOGRAPHY AND NETWORK SECURITY

UNIT 1 NOTES

PREPARED BY

R.CYNTHIA PRIYADHARSHINI

AP/IT/SREC

UNIT – I INTRODUCTION & NUMBER THEORY

Services, Mechanisms and attacks-the OSI security architecture-Network security model-Classical Encryption techniques (Symmetric cipher model, substitution techniques, transposition techniques, steganography).FINITE FIELDS AND NUMBER THEORY: Groups, Rings, Fields-Modular arithmetic-Euclid's algorithm-Finite fields-Polynomial Arithmetic -Prime numbers-Fermat's and Euler's theorem-Testing for primality -The Chinese remainder theorem- Discrete logarithms.

COMPUTER SECURITY CONCEPTS

Computer Security

The protection afforded to an automated information system in order to attain the applicable objectives of preserving the integrity, availability, and confidentiality of information system resources (includes hardware, software, firmware, information / data, and telecommunications)

Confidentiality

- Data confidentiality
 - Assures that private or confidential information is not made available or disclosed to unauthorized
- Privacy
 - Assures that individuals control or influence what information related to them may be collected and stored and by whom and to whom that information may be disclosed.

Integrity

- Data integrity
 - Assures that information and programs are changed only in a specified and authorized manner.
- System integrity
 - Assures that a system performs its intended function in an unimpaired manner, free from deliberate or inadvertent unauthorized manipulation of the system.

Availability

- Assures that systems work promptly and service is not denied to authorized users.

CIA Triad

Confidentiality

- Preserving authorized restrictions on information access and disclosure, including means for protecting personal privacy and proprietary information.
- A loss of confidentiality is the unauthorized disclosure of information.

Integrity

- Guarding against improper information modification or destruction, including ensuring information nonrepudiation and authenticity.
- A loss of integrity is the unauthorized modification or destruction of information.

Availability

- Ensuring timely and reliable access to and use of information
- A loss of availability is the disruption of access to or use of information or an information system.

Authenticity

- The property of being genuine and being able to be verified and trusted

Accountability

- The security goal that generates the requirement for actions of an entity to be traced uniquely to that entity

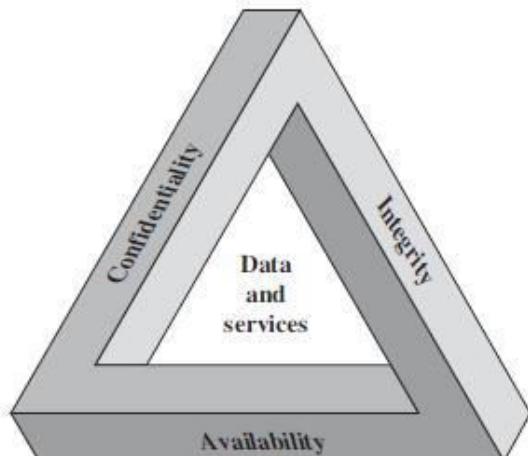


Figure 1.1 The Security Requirements Triad

The OSI Security Architecture

- ITU-T Recommendation X.800, Security Architecture for OSI, defines such a systematic approach
- The OSI security architecture focuses on security attacks, mechanisms, and services.

Security attack

- Any action that compromises the security of information owned by an organization.

Security mechanism

- A process (or a device) that is designed to detect, prevent, or recover from a security attack.

Security service

- A processing or communication service that enhances the security of the data processing systems and the information transfers of an organization
- The services are intended to counter security attacks, and they make use of one or more security mechanisms to provide the service

Security Attacks

- means of classifying security attacks, used both in X.800 and RFC 2828
- A passive attack attempts to learn or make use of information but does not affect system resources.
- An active attack attempts to alter system resources or affect their operation.

Passive Attacks

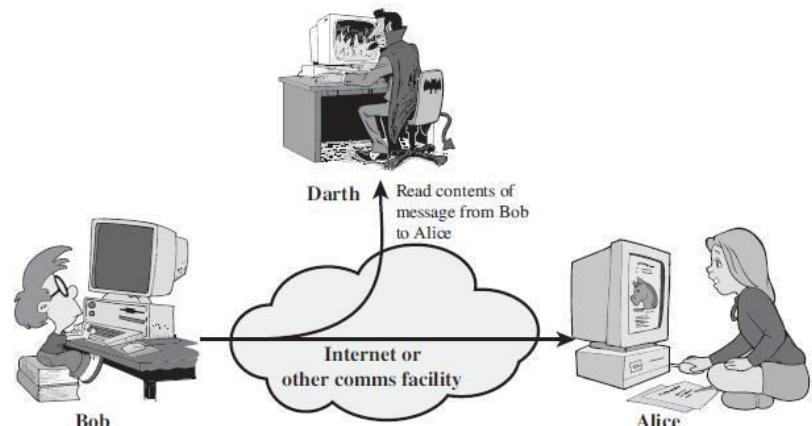
- in the nature of eavesdropping on, or monitoring of, transmissions.
- The goal is to obtain information that is being transmitted.
- very difficult to detect, because they do not involve any alteration of the data
- feasible to prevent the success of these attacks, usually by means of encryption
- emphasis in dealing with passive attacks is on prevention rather than detection

Two types of passive attacks

- Release of message contents
- Traffic analysis.

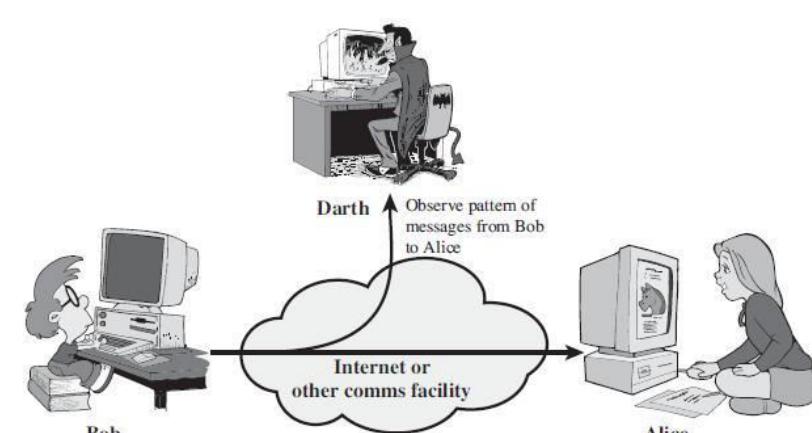
Release Of Message Contents

- A telephone conversation, an electronic mail message, and a transferred file may contain sensitive or confidential information
- prevent an opponent from learning the contents of these transmissions



Traffic Analysis

- observe the pattern of these messages
- The opponent could determine the location and identity of communicating hosts and could observe the frequency and length of messages being exchanged.
- This information might be useful in guessing the nature of the communication that was taking place



Active Attacks

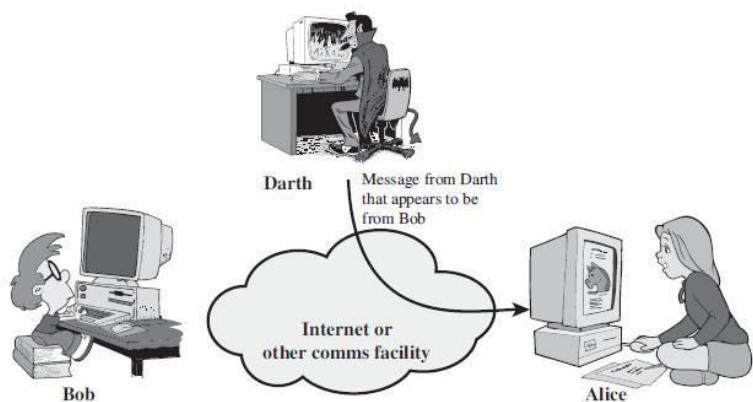
- Active attacks involve some modification of the data stream or the creation of a false stream
- detect and to recover from any disruption or delays caused by them
- can be subdivided into four categories:
 - masquerade,
 - replay,
 - modification of messages
 - denial of service

Masquerade

- one entity pretends to be a different entity
- usually includes one of the other forms of active attack

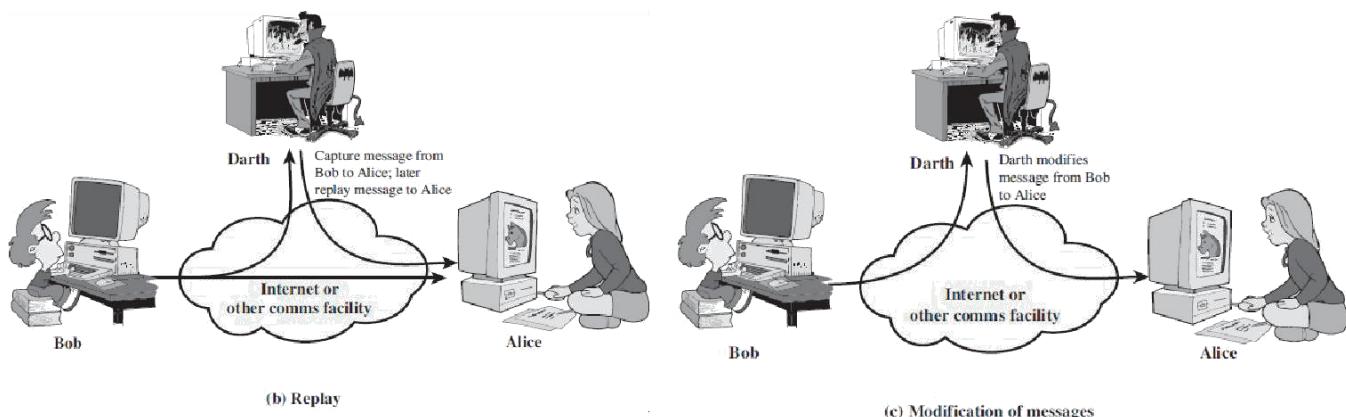
Example

- authentication sequences can be captured and replayed after a valid authentication sequence



Replay

- passive capture of a data unit and its subsequent retransmission to produce an unauthorized effect



Modification Of Messages

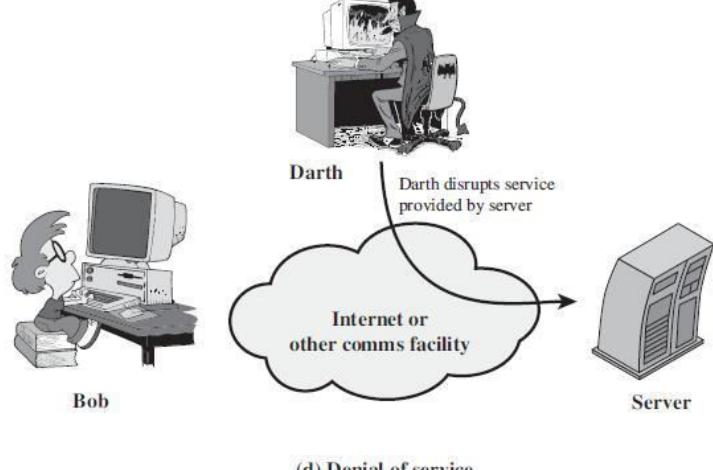
- some portion of a legitimate message is altered, or that messages are delayed or reordered, to produce an unauthorized effect

Example

- a message meaning “Allow John Smith to read confidential file accounts” is modified to mean “Allow Fred Brown to read confidential file accounts.”

Denial Of Service

- prevents or inhibits the normal use or management of communications facilities
- may have a specific target; for example, an entity may suppress all messages directed to a particular destination
- disruption of an entire network, either by disabling the network or by overloading it with messages so as to degrade performance



Security Services in X.800

- X.800 defines a security service as a service that is provided by a protocol layer of communicating open systems and that ensures adequate security of the systems or of data transfers.
- RFC 2828, defines as a processing or communication service that is provided by a system to give a specific kind of protection to system resources;
 - security services implement security policies and are implemented by security mechanisms.

X.800

- divides these services into five categories and fourteen specific services

Authentication

- The assurance that the communicating entity is the one that it claims to be
- Two types
 - Peer Entity Authentication
 - Data-Origin Authentication

Access control

- The prevention of unauthorized use of a resource

Data confidentiality

- The protection of data from unauthorized disclosure.
- Four Types
 - Connection Confidentiality
 - Connectionless Confidentiality
 - Selective-Field Confidentiality
 - Traffic-Flow Confidentiality

Data integrity

- The assurance that data received are exactly as sent by an authorized entity (i.e., contain no modification, insertion, deletion, or replay).
- Five types
 - Connection Integrity with Recovery
 - Connection Integrity without Recovery
 - Selective-Field Connection Integrity
 - Connectionless Integrity
 - Selective-Field Connectionless Integrity

Nonrepudiation

- Provides protection against denial by one of the entities involved in a communication of having participated in all or part of the communication
- Two types
 - Nonrepudiation, Origin
 - Nonrepudiation, Destination

Security Mechanisms in X.800.

- feature designed to detect, prevent, or recover from a security attack
- no single mechanism that will support all services required

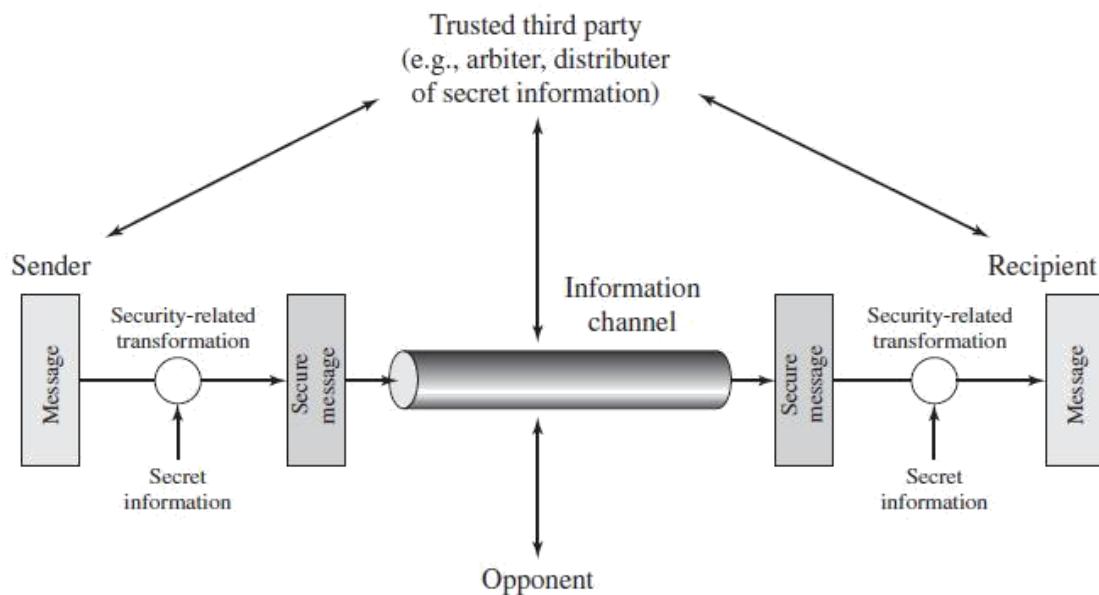
Specific security mechanisms:

- those that are implemented in a specific protocol layer, such as TCP or an application-layer protocol
- encipherment, digital signatures, access controls, data integrity, authentication exchange, traffic padding, routing control, notarization

pervasive security mechanisms:

- trusted functionality, security labels, event detection, security audit trails, security recovery
- those that are not specific to any particular protocol layer or security service

Model for Network Security



- A message is to be transferred from one party to another across some sort of Internet service.
- The two parties, who are the principals in this transaction, must cooperate for the exchange to take place.
- A logical information channel is established by defining a route through the Internet from source to destination and by the cooperative use of communication protocols (e.g., TCP/IP) by the two principals

All the techniques for providing security have two components:

- A security-related transformation on the information to be sent.
 - Examples: encryption of the message, addition of a code based on the contents
- Some secret information shared by the two principals, unknown to the opponent
 - Example: encryption key used in conjunction with the transformation

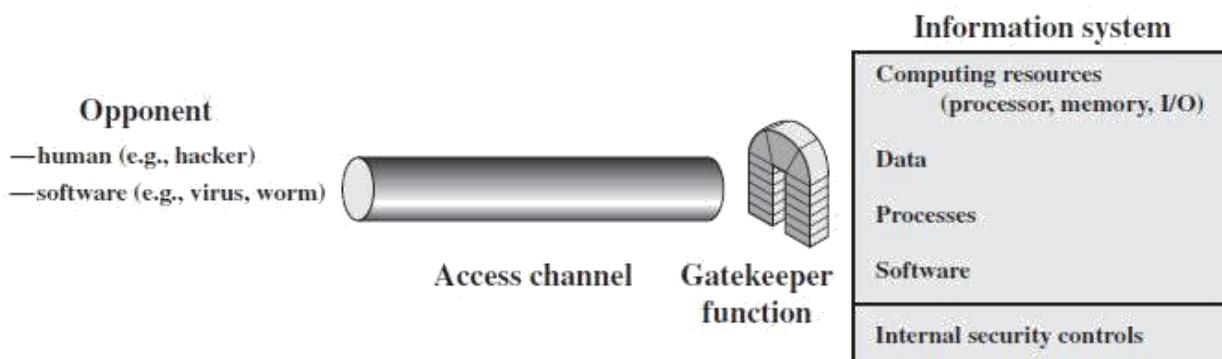
A trusted third party may be needed to achieve secure transmission.

- for distributing the secret information to the two principals
- to arbitrate disputes between the two principals concerning the authenticity of a message transmission

Four basic tasks in designing a particular security service:

1. Design an algorithm for performing the security-related transformation
 - such that an opponent cannot defeat its purpose.
2. Generate the secret information to be used with the algorithm.
3. Develop methods for the distribution and sharing of the secret information.
4. Specify a protocol to be used by the two principals that makes use of the security algorithm and the secret information to achieve a particular security service

Network Access Security Model



- protecting an information system from unwanted access from hacker, intruder
- hacker who, with no malign intent, simply gets satisfaction from breaking and entering a computer system.
- intruder can be a disgruntled employee who wishes to do damage or a criminal who seeks to exploit computer assets for financial gain
- placement in a computer system of logic that exploits vulnerabilities in the system and that can affect application programs as well as utility programs, such as editors and compilers
 - Two kinds of threats:
 - **Information access threats:** Intercept or modify data on behalf of users who should not have access
 - **Service threats:** Exploit service flaws in computers to inhibit use by legitimate users
 - Examples: Viruses and worms, spread using disks & inserted over network

Classical Encryption Techniques

- Symmetric Cipher Model
 - Cryptanalysis and Brute-Force Attack
- Substitution Techniques
 - Caesar Cipher
 - Monoalphabetic Ciphers
 - Playfair Cipher
 - Hill Cipher
 - Polyalphabetic Ciphers
 - One-Time Pad
- Transposition Techniques
- Rotor Machines
- Steganography

Introduction

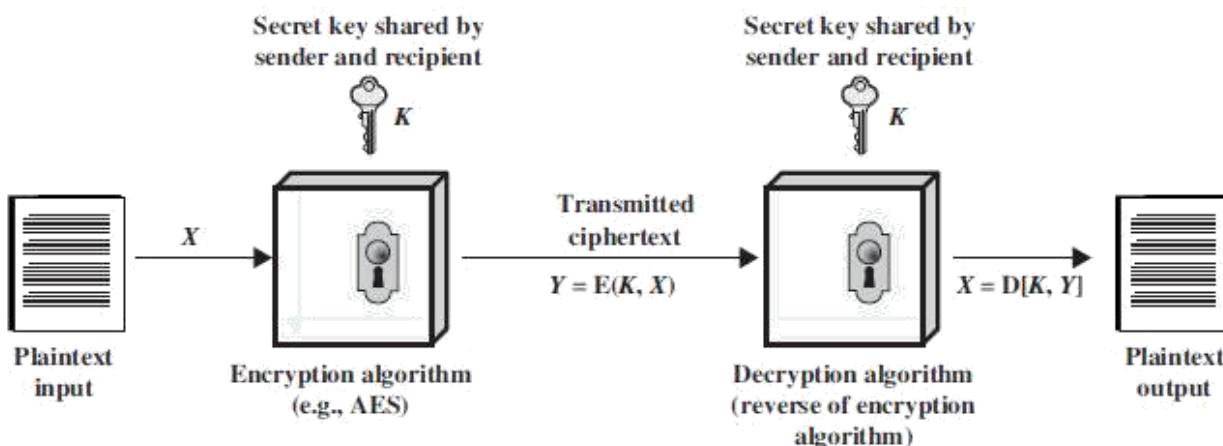
- **Symmetric encryption** is a form of cryptosystem in which encryption and decryption are performed using the **same key**. It is also known as **conventional encryption**.
- Symmetric encryption transforms plaintext into ciphertext using a secret key and an encryption algorithm. Using the same key and a decryption algorithm, the plaintext is recovered from the ciphertext.
- The two types of attack on an encryption algorithm are **cryptanalysis**, based on properties of the encryption algorithm, and **brute-force**, which involves trying all possible keys.
- Traditional (precomputer) symmetric ciphers use substitution and/or transposition techniques. Substitution techniques map plaintext elements (characters, bits) into ciphertext elements. Transposition techniques systematically transpose the positions of plaintext elements.
- Rotor machines are sophisticated precomputer hardware devices that use substitution techniques.
- Steganography is a technique for hiding a secret message within a larger one in such a way that others cannot discern the presence or contents of the hidden message.
- An original message is known as the **plaintext**, while the coded message is called the **ciphertext**.
- The process of converting from plaintext to ciphertext is known as **enciphering or encryption**; restoring the plaintext from the ciphertext is **deciphering or decryption**.
- The many schemes used for encryption constitute the area of study known as **cryptography**. Such a scheme is known as a **cryptographic system or a cipher**.
- Techniques used for deciphering a message without any knowledge of the enciphering details fall into the area of cryptanalysis. **Cryptanalysis** is what the layperson calls “breaking the code.” The areas of cryptography and cryptanalysis together are called **cryptology**

Symmetric Cipher Model

A symmetric encryption scheme has five ingredients

- Plaintext
- Encryption algorithm
 - performs various substitutions and transformations
- Secret key
 - another input to the encryption algorithm
 - a value independent of the plaintext and of the algorithm
- Ciphertext
 - For a given message, two different keys will produce two different ciphertexts
- Decryption algorithm
 - encryption algorithm run in reverse

Simplified Model of Symmetric Encryption



Two requirements for secure use of conventional / symmetric encryption

- need a strong encryption algorithm
 - The opponent should be unable to decrypt ciphertext or discover the key even if he or she is in possession of a number of ciphertexts together with the plaintext that produced each ciphertext
- Sender and receiver must have obtained copies of the secret key in a secure fashion and must keep the key secure.
 - If someone can discover the key and knows the algorithm, all communication using this key is readable
 - do not need to keep the algorithm secret; we need to keep only the key secret
 - the principal security problem is maintaining the secrecy of the key

Model of Symmetric Cryptosystem

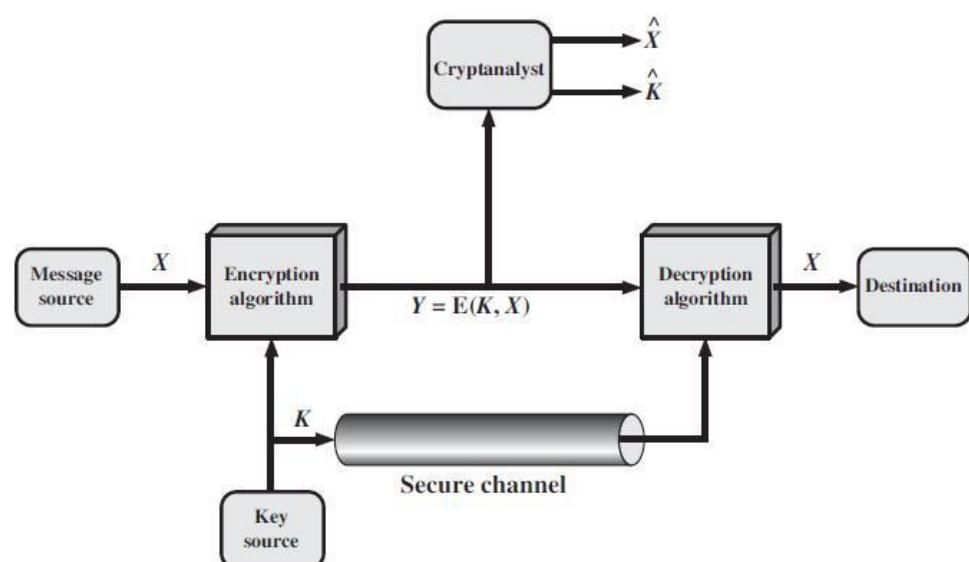
Plain Text: $X = [X_1, X_2, \dots, X_M]$

Key: $K = [K_1, K_2, \dots, K_J]$

Cipher text $Y = [Y_1, Y_2, \dots, Y_N]$

$$Y = E(K, X)$$

$$X = D(K, Y)$$



Cryptanalysis and Brute-Force Attack

Cryptanalysis

- Cryptanalytic attacks rely on the nature of the algorithm plus perhaps some knowledge of the general characteristics of the plaintext or even some sample plaintext–ciphertext pairs.
- This type of attack exploits the characteristics of the algorithm to attempt to deduce a specific plaintext or to deduce the key being used.
- various types of cryptanalytic attacks based on the amount of information known to the cryptanalyst

Type of Attack	Known to Cryptanalyst
<ul style="list-style-type: none"> • Ciphertext Only • Known Plaintext • Chosen Plaintext • Chosen Ciphertext • Chosen Text 	<ul style="list-style-type: none"> • Encryption algorithm • Ciphertext • Plaintext message chosen by cryptanalyst, together with its corresponding ciphertext generated with the secret key • Ciphertext chosen by cryptanalyst, together with its corresponding decrypted plaintext generated with the secret key

Two schemes

- **unconditionally secure**
 - if the ciphertext generated by the scheme does not contain enough information to determine uniquely the corresponding plaintext, no matter how much ciphertext is available
- **computationally secure**
 - meets either of the following criteria:
 - 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.

Brute-force attack

- The attacker tries every possible key on a piece of ciphertext until an intelligible translation into plaintext is obtained.
- On average, half of all possible keys must be tried to achieve success.

Cryptographic systems characterization

Three independent dimensions

- The type of operations used for transforming plaintext to ciphertext.
 - **substitution**
 - each element is mapped into another element
 - **transposition**
 - elements are rearranged
 - product systems, involve multiple stages of substitutions and transpositions
- The number of keys used
 - If both sender and receiver use the same key, the system is referred to as **symmetric**, single-key, secret-key, or conventional encryption.
 - If the sender and receiver use different keys, the system is referred to as **asymmetric**, two-key, or public-key encryption
- The way in which the plaintext is processed.
 - A **block cipher** processes the input one block of elements at a time, producing an output block for each input block.
 - A **stream cipher** processes the input elements continuously, producing output one element at a time, as it goes along

Substitution Techniques

- A substitution technique is one in which the letters of plaintext are replaced by other letters or by numbers or symbols
- If the plaintext is viewed as a sequence of bits, then substitution involves replacing plaintext bit patterns with ciphertext bit patterns

Julius Caesar Cipher

- replacing each letter of the alphabet with the letter standing three places further down the alphabet
- alphabet is wrapped around, so that the letter following Z is A

can define transformation as:

a b c d e f g h i j k l m n o p q r s t u v w x y z D
E F G H I J K L M N O P Q R S T U V W X Y Z A B C

mathematically give each letter a number

a b c d e f g h i j k l m n o p q r s t u v w x y z
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25

then have Caesar cipher as:

$$\begin{aligned}c &= E(p) = (p + k) \bmod (26) \\&= D(c) = (c - k) \bmod (26)\end{aligned}$$

Cryptanalysis of Caesar Cipher

- only have 26 possible ciphers
- A maps to A,B,..Z
- could simply try each in turn
- a brute force search
- given ciphertext, just try all shifts of letters
- do need to recognize when have plaintext

Monoalphabetic Ciphers

- rather than just shifting the alphabet shuffle (jumble) the letters arbitrarily
- each plaintext letter maps to a different random ciphertext letter
- hence key is 26 letters long
- the “cipher” line can be any permutation of the 26 alphabetic characters, then there are $26!$ or greater than 4×10^{26} possible keys.
- This is 10 orders of magnitude greater than the key space for DES and would seem to eliminate brute-force techniques for cryptanalysis
- Monoalphabetic ciphers are easy to break because they reflect the frequency data of the original alphabet
- A countermeasure is to provide multiple substitutes, known as **homophones**, for a single letter.
- For example, the letter e could be assigned a number of different cipher symbols, such as 16, 74, 35, and 21, with each homophone assigned to a letter in rotation or randomly

Language Redundancy and Cryptanalysis

- human languages are redundant
- eg "th lrd s m shphrd shll nt wnt"
- letters are not equally commonly used
- in English E is by far the most common letter
- followed by T,R,N,I,O,A,S
- other letters like Z,J,K,Q,X are fairly rare
- have tables of single, double & triple letter frequencies for various languages
- two-letter combinations, known as **digrams** (ex: th)

Playfair Cipher

- best-known multiple-letter encryption cipher
- treats digrams in the plaintext as single units and translates these units into ciphertext digrams

Playfair Key Matrix

- 5×5 matrix of letters constructed using a keyword
- filling in the letters of the keyword (minus duplicates) from left to right and from top to bottom,
- filling in the remainder matrix with the remaining letters in alphabetic order.
- The letters I and J count as one letter
- Example matrix using the keyword MONARCHY

M	O	N	A	R
C	H	Y	B	D
E	F	G	I/J	K
L	P	Q	S	T
U	V	W	X	Z

Plaintext is encrypted two letters at a time, according to the following rules

- Repeating plaintext letters that are in the same pair are separated with a filler letter, such as x,
 - Ex: balloon would be treated as ba lx lo on.
- Two plaintext letters that fall in the same row of the matrix are each replaced by the letter to the right, with the first element of the row circularly following the last.
 - Ex: ar is encrypted as RM.
- Two plaintext letters that fall in the same column are each replaced by the letter beneath, with the top element of the column circularly following the last.
 - Ex: mu is encrypted as CM.
- Otherwise, each plaintext letter in a pair is replaced by the letter that lies in its own row and the column occupied by the other plaintext letter.
 - Ex: hs becomes BP and ea becomes IM (or JM, as the encipherer wishes)

Example

Given the key MONARCHY apply Play fair cipher to plain text “FACTIONALISM”

Solution

- (p) FA CT IO NA LI SM
 (c) IO DL FA AR SE LA
 (d) FA CT IO NA LI SM

Security of Playfair Cipher

- security much improved over monoalphabetic since have $26 \times 26 = 676$ digrams
- would need a 676 entry frequency table to analyse and correspondingly more ciphertext
- was widely used for many years eg. by US & British military in WW1
- it can be broken, given a few hundred letters since still has much of plaintext structure

Hill Cipher

Findin

$$\begin{pmatrix} k_{11} & k_{12} & k_{13} \\ k_{21} & k_{22} & k_{23} \\ k_{31} & k_{32} & k_{33} \end{pmatrix}$$

$$\text{3x3 Matrix } \begin{pmatrix} k_1 & k_2 & k_3 \\ k_4 & k_5 & k_6 \\ k_7 & k_8 & k_9 \end{pmatrix} \quad \text{3x3 Matrix } \begin{pmatrix} k_{11} & k_{12} & k_{13} \\ k_{21} & k_{22} & k_{23} \\ k_{31} & k_{32} & k_{33} \end{pmatrix} \quad |A| = k_{11}k_{22}k_{33} - k_{11}k_{23}k_{32} - k_{12}k_{21}k_{33} + k_{12}k_{23}k_{31} + k_{13}k_{21}k_{32} - k_{13}k_{22}k_{31}$$

Example:

$$\det \begin{pmatrix} 5 & 8 \\ 17 & 3 \end{pmatrix} = (5 \times 3) - (8 \times 17) = -121 \bmod 26 = 9$$

We can show that $9^{-1} \bmod 26 = 3$, because $9 \times 3 = 27 \bmod 26 = 1$ (see Chapter 4 or Appendix E). Therefore, we compute the inverse of \mathbf{A} as

$$\mathbf{A} = \begin{pmatrix} 5 & 8 \\ 17 & 3 \end{pmatrix}$$

$$\mathbf{A}^{-1} \bmod 26 = 3 \begin{pmatrix} 3 & -8 \\ -17 & 5 \end{pmatrix} = 3 \begin{pmatrix} 3 & 18 \\ 9 & 5 \end{pmatrix} = \begin{pmatrix} 9 & 54 \\ 27 & 15 \end{pmatrix} = \begin{pmatrix} 9 & 2 \\ 1 & 15 \end{pmatrix}$$

The Hill algorithm

- This encryption algorithm takes m successive plaintext letters and substitutes for them m ciphertext letters.
- The substitution is determined by m linear equations in which each character is assigned a numerical value ($a = 0, b = 1, \dots, z = 25$)
- For $m = 3$, the system can be described as

$$c_1 = (k_{11}p_1 + k_{12}p_2 + k_{13}p_3) \bmod 26$$

$$c_2 = (k_{21}p_1 + k_{22}p_2 + k_{23}p_3) \bmod 26$$

$$c_3 = (k_{31}p_1 + k_{32}p_2 + k_{33}p_3) \bmod 26$$

This can be expressed in terms of row vectors and matrices:

$$(c_1 \ c_2 \ c_3) = (p \ p_2 \ p_3) \begin{pmatrix} k_{11} & k_{12} & k_{13} \\ k_{21} & k_{22} & k_{23} \\ k_{31} & k_{32} & k_{33} \end{pmatrix} \bmod 26$$

or

$$\mathbf{C} = \mathbf{PK} \bmod 26$$

- where C and P are row vectors of length 3 representing the plaintext and ciphertext, and K is a 3×3 matrix representing the encryption key.
- Operations are performed mod 26.
- In general terms, the Hill system can be expressed as

$$\mathbf{C} = \mathbf{E(K, P)} = \mathbf{PK} \bmod 26$$

$$\mathbf{P} = \mathbf{D(K, C)} = \mathbf{CK}^{-1} \bmod 26 = \mathbf{PKK}^{-1} = \mathbf{P}$$

Example

Encrypt the message “meet me at the usual place at ten rather than eight o'clock” using the Hill cipher with the key (\cdot) . Show your calculations and the result. Show the calculations for the corresponding decryption of the ciphertext to recover the original plaintext.

1) mathematically give each letter a number

a b c d e f g h i j k l m n o p q r s t u v w x y z

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25

2) 1st pair from plain text “me” $\Rightarrow \begin{pmatrix} 12 \\ 4 \end{pmatrix}$

$$\begin{pmatrix} 9 & 4 \\ 5 & 7 \end{pmatrix} \begin{pmatrix} 12 \\ 4 \end{pmatrix} \Rightarrow \begin{pmatrix} 9 \times 12 + 4 \times 4 \\ 5 \times 12 + 7 \times 4 \end{pmatrix} = \begin{pmatrix} 124 \\ 88 \end{pmatrix} \Rightarrow \bmod 26 \Rightarrow \begin{pmatrix} 20 \\ 10 \end{pmatrix} \Rightarrow \begin{pmatrix} u \\ k \end{pmatrix}$$

3) 2nd pair from plain text “et”

$$\begin{pmatrix} 9 & 4 \\ 5 & 7 \end{pmatrix} \begin{pmatrix} 4 \\ 19 \end{pmatrix} \Rightarrow \begin{pmatrix} 9 \times 4 + 4 \times 19 \\ 5 \times 4 + 7 \times 19 \end{pmatrix} = \begin{pmatrix} 112 \\ 153 \end{pmatrix} \Rightarrow \bmod 26 \Rightarrow \begin{pmatrix} 8 \\ 23 \end{pmatrix} \Rightarrow \begin{pmatrix} i \\ x \end{pmatrix}$$

4) Cipher text for "meet" is "ukix"

5) To get plain text from cipher text, we need to find the inverse of K

6) $|A| = (9 \times 7 - 5 \times 4) \Rightarrow 43$

$$7) \text{Adj}(A) \Rightarrow \begin{pmatrix} 7 & -4 \\ -5 & 9 \end{pmatrix} \Rightarrow \frac{1}{43} \begin{pmatrix} 7 & -4 \\ -5 & 9 \end{pmatrix} (\because 43 \% 26 = 17)$$

8) Find the multiplier for 17, using $17 \times X = 1 \pmod{26} \Rightarrow X = 23$

$$9) \begin{pmatrix} 161 & -92 \\ -115 & 207 \end{pmatrix} \Rightarrow \pmod{26} \Rightarrow \begin{pmatrix} 5 & -14 \\ -11 & 25 \end{pmatrix} \Rightarrow \begin{pmatrix} 5 & 12 \\ 15 & 25 \end{pmatrix} (\because \text{Add 26 for negative values})$$

10) $P = CK^{-1}$ \Rightarrow For the ciper text of "uk",

$$\begin{pmatrix} 5 & 12 \\ 15 & 25 \end{pmatrix} \begin{pmatrix} 20 \\ 10 \end{pmatrix} = \begin{pmatrix} 5 \times 20 + 12 \times 10 \\ 15 \times 20 + 25 \times 10 \end{pmatrix} \Rightarrow \begin{pmatrix} 220 \\ 550 \end{pmatrix} \pmod{26} \Rightarrow \begin{pmatrix} 12 \\ 4 \end{pmatrix} = \begin{pmatrix} m \\ e \end{pmatrix}$$

Hence the plain text is "me"

Polyalphabetic Ciphers

- use different monoalphabetic substitutions as one proceeds through the plaintext message.
- improve security using multiple cipher alphabets
- make cryptanalysis harder with more alphabets to guess and flatter frequency distribution
- general name for this approach is **polyalphabetic substitution cipher**
- has the following features in common:
 - A set of related monoalphabetic substitution rules is used.
 - A key determines which particular rule is chosen for a given transformation.

One-Time Pad

- improvement to the Vernam cipher that yields the ultimate in security
- using a random key that is as long as the message, so that the key need not be repeated
- the key is to be used to encrypt and decrypt a single message, and then is discarded.
- Each new message requires a new key of the same length as the new message

Example

ciphertext: ANKYODKYUREPFJBYOJDSPLREYIUNOFDOIUERFPLUYTS

key: pxlmvmsydfuyrvzwc tnlebnecvgdupahfzzlmnyih

plaintext: mr mustard with the candlestick in the hall

ciphertext: ANKYODKYUREPFJBYOJDSPLREYIUNOFDOIUERFPLUYTS

key: mfugpmiydgaxgoufhklllmlhsqdqogtewbqfggyovuhwt

plaintext: miss scarlet with the knife in the library

two fundamental difficulties

- problem of making large quantities of random keys
- problem of key distribution and protection

Transposition Techniques

A very different kind of mapping is achieved by performing some sort of permutation on the plaintext letters

Rail Fence Technique

The simplest such cipher is the rail fence technique, in which the plaintext is written down as a sequence of diagonals and then read off as a sequence of rows.

For example, to encipher the message "meet me after the toga party" with a rail fence of depth 2, we write the following

```
m e m a t r h t g p r y  
e t e f e t e o a a t
```

The encrypted message is

MEMATRHTGPRYETEFETEOAAT

Pure Transposition Cipher

write the message in a rectangle, row by row, and read the message off, column by column, but permute the order of the columns.

The order of the columns then becomes the key to the algorithm

Example

Key: 4 3 1 2 5 6 7

Plaintext: a t t a c k p
o s t p o n e
d u n t i l t
w o a m x y z

Ciphertext: TTNAAPMTSUOAODWCOIXKNLYPETZ

Double Transposition

performing more than one stage of transposition

Example

if the foregoing message is reencrypted using the same algorithm

Key: 4 3 1 2 5 6 7

Input: t t n a a p t
m t s u o a o
d w c o i x k
n l y p e t z

Output: NSCYAUOPTTWLTMDNAOIEPAXTOKZ

This is a much less structured permutation and is much more difficult to cryptanalyze

Rotor Machines (Skip)

The machine consists of a set of independently rotating cylinders through which electrical pulses can flow.

Each cylinder has 26 input pins and 26 output pins, with internal wiring that connects each input pin to a unique output pin

Steganography

A plaintext message may be hidden in one of two ways.

- The methods of steganography conceal the existence of the message
- The methods of cryptography render the message unintelligible to outsiders
 - by various transformations of the text

Various ways to conceal the message

arrangement of words or letters within an apparently innocuous text spells out the real message

Character marking

Selected letters of printed or typewritten text are overwritten in pencil. The marks are ordinarily not visible unless the paper is held at an angle to bright light.

Invisible ink

A number of substances can be used for writing but leave no visible trace until heat or some chemical is applied

Pin punctures

Small pin punctures on selected letters are ordinarily not visible unless the paper is held up in front of a light.

Typewriter correction ribbon

Used between lines typed with a black ribbon, the results of typing with the correction tape are visible only under a strong light

hiding a message by using the least significant bits of frames on a CD

- the Kodak Photo CD format's maximum resolution is 2048 by 3072 pixels, with each pixel containing 24 bits of RGB color information.
- The least significant bit of each 24-bit pixel can be changed without greatly affecting the quality of the image
- Thus you can hide a 2.3-megabyte message in a single digital snapshot

Number of drawbacks

- lot of overhead to hide a relatively few bits of information
- once the system is discovered, it becomes virtually worthless
- the insertion method depends on some sort of key
 - Alternatively, a message can be first encrypted and then hidden using steganography

Advantage of steganography

- can be employed by parties who have something to lose should the fact of their secret communication be discovered
- Encryption flags traffic as important or secret or may identify the sender or receiver as someone with something to hide

FINITE FIELDS AND NUMBER THEORY

Number Theory concepts

Divisibility and The Division Algorithm

Divisibility

- We say that a nonzero b divides a if $a = mb$ for some m , where a, b and m are integers.
- That is, b divides a , if there is no remainder on division.
- The notation $b|a$ is commonly used to mean b divides a .
- If $b|a$, we say that b is a divisor of a .

The positive divisors of 24 are 1, 2, 3, 4, 6, 8, 12, and 24.
 $13|182; -5|30; 17|289; -3|33; 17|0$

Properties of divisibility for integers

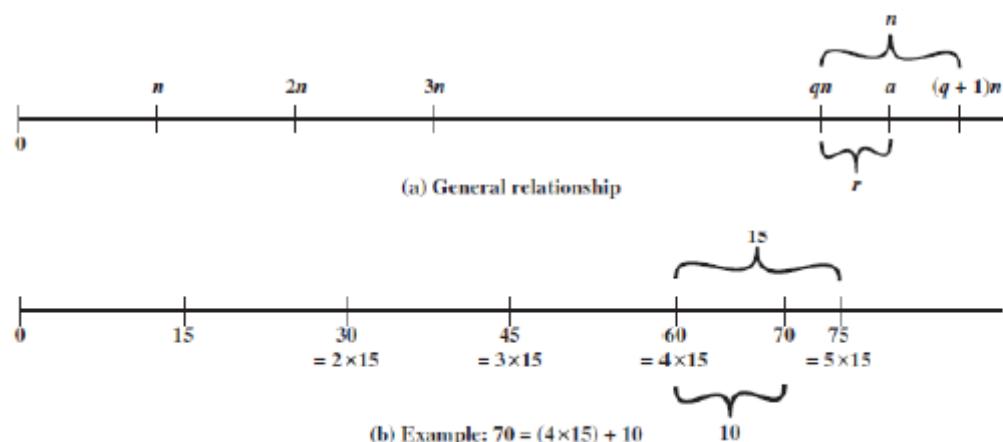
- If $a|1$, then $a = \pm 1$.
- If $a|b$ and $b|a$, then $a = \pm b$.
- Any $b \neq 0$ divides 0.
- If $a|b$ and $b|c$, then $a|c$: $11|66$ and $66|198 = 11|198$
- If $b|g$ and $b|h$, then $b|(mg + nh)$ for arbitrary integers m and n .

The Division Algorithm

Given any positive integer n and any nonnegative integer a , if we divide a by n , we get an integer quotient and an integer remainder r that obey the following relationship: The remainder r is often referred to as a residue.

$$a = qn + r \quad 0 \leq r < n; q = \lfloor a/n \rfloor$$

where $\lfloor x \rfloor$ is the largest integer less than or equal to x .



The Euclidean Algorithm

- Simple procedure for determining the greatest common divisor of two positive integers.
- Two integers are relatively prime if their only common positive integer factor is 1

Greatest Common Divisor

- largest integer that divides both a and b
- $\gcd(0, 0) = 0$.

the positive integer c is said to be the greatest common divisor of a and b if

1. c is a divisor of a and of b
2. Any divisor of a and b is a divisor of c

$$\gcd(a, b) = \max[k, \text{such that } k|a \text{ and } k|b] \quad \gcd(a, b) = \gcd(|a|, |b|).$$

$$\gcd(60, 24) = \gcd(60, -24) = 12$$

- a and b are relatively prime if $\gcd(a, b) = 1$

Example:

8 and 15 are relatively prime because the positive divisors of 8 are 1, 2, 4, and 8, and the positive divisors of 15 are 1, 3, 5, and 15. So 1 is the only integer on both lists.

Steps

$$\begin{array}{ll} a = q_1b + r_1 & 0 < r_1 < b \\ b = q_2r_1 + r_2 & 0 < r_2 < r_1 \\ r_1 = q_3r_2 + r_3 & 0 < r_3 < r_2 \\ \vdots & \vdots \\ \vdots & \vdots \\ \vdots & \vdots \\ r_{n-2} = q_n r_{n-1} + r_n & 0 < r_n < r_{n-1} \\ r_{n-1} = q_{n+1} r_n + 0 & \\ d = \gcd(a, b) = r_n & \end{array}$$

Example

To find $d = \gcd(a, b) = \gcd(1160718174, 316258250)$			
$a = q_1b + r_1$	$1160718174 = 3 \times 316258250 + 211943424$	$d = \gcd(316258250, 211943424)$	
$b = q_2r_1 + r_2$	$316258250 = 1 \times 211943424 + 104314826$	$d = \gcd(211943424, 104314826)$	
$r_1 = q_3r_2 + r_3$	$211943424 = 2 \times 104314826 + 3313772$	$d = \gcd(104314826, 3313772)$	

Dividend	Divisor	Quotient	Remainder
$a = 1160718174$	$b = 316258250$	$q_1 = 3$	$r_1 = 211943424$
$b = 316258250$	$r_1 = 211943424$	$q_2 = 1$	$r_2 = 104314826$
$r_1 = 211943424$	$r_2 = 104314826$	$q_3 = 2$	$r_3 = 3313772$

Euclidean Algorithm Revisited

- For any nonnegative integer a and any positive integer b ,
- $\gcd(a, b) = \gcd(b, a \bmod b)$
- Example: $\gcd(55, 22) = \gcd(22, 55 \bmod 22) = \gcd(22, 11) = 11$

Recursive function.

```
Euclid(a,b)
    if (b=0) then return a;
    else return Euclid(b, a mod b);
```

The Extended Euclidean Algorithm

calculate the greatest common divisor but also two additional integers and that satisfy the following equation

$$ax + by = d = \gcd(a, b)$$

x and y will have opposite signs

(4.3), and we assume that at each step i we can find integers x_i and y_i that satisfy $r_i = ax_i + by_i$. We end up with the following sequence.

$$\begin{array}{ll} a = q_1b + r_1 & r_1 = ax_1 + by_1 \\ b = q_2r_1 + r_2 & r_2 = ax_2 + by_2 \\ r_1 = q_3r_2 + r_3 & r_3 = ax_3 + by_3 \\ \vdots & \vdots \\ r_{n-2} = q_n r_{n-1} + r_n & r_n = ax_n + by_n \\ r_{n-1} = q_{n+1} r_n + 0 & \end{array}$$

we can rearrange terms to write

$$r_i = r_{i-2} - r_{i-1} q_i$$

Also, in rows $i - 1$ and $i - 2$, we find the values

$$r_{i-2} = ax_{i-2} + by_{i-2} \quad \text{and} \quad r_{i-1} = ax_{i-1} + by_{i-1}$$

Substituting into Equation (4.8), we have

$$\begin{aligned} r_i &= (ax_{i-2} + by_{i-2}) - (ax_{i-1} + by_{i-1})q_i \\ &= a(x_{i-2} - q_i x_{i-1}) + b(y_{i-2} - q_i y_{i-1}) \end{aligned}$$

But we have already assumed that $r_i = ax_i + by_i$. Therefore,

$$x_i = x_{i-2} - q_i x_{i-1} \quad \text{and} \quad y_i = y_{i-2} - q_i y_{i-1}$$

Let us now look at an example with relatively large numbers to see the power of this algorithm:

To find $d = \gcd(a, b) = \gcd(1160718174, 316258250)$			
$a = q_1b + r_1$	$1160718174 = 3 \times 316258250 + 211943424$	$d = \gcd(316258250, 211943424)$	
$b = q_2r_1 + r_2$	$316258250 = 1 \times 211943424 + 104314826$	$d = \gcd(211943424, 104314826)$	
$r_1 = q_3r_2 + r_3$	$211943424 = 2 \times 104314826 + 3313772$	$d = \gcd(104314826, 3313772)$	
$r_2 = q_4r_3 + r_4$	$104314826 = 31 \times 3313772 + 1587894$	$d = \gcd(3313772, 1587894)$	
$r_3 = q_5r_4 + r_5$	$3313772 = 2 \times 1587894 + 137984$	$d = \gcd(1587894, 137984)$	
$r_4 = q_6r_5 + r_6$	$1587894 = 11 \times 137984 + 70070$	$d = \gcd(137984, 70070)$	
$r_5 = q_7r_6 + r_7$	$137984 = 1 \times 70070 + 67914$	$d = \gcd(70070, 67914)$	
$r_6 = q_8r_7 + r_8$	$70070 = 1 \times 67914 + 2156$	$d = \gcd(67914, 2156)$	
$r_7 = q_9r_8 + r_9$	$67914 = 31 \times 2156 + 1078$	$d = \gcd(2156, 1078)$	
$r_8 = q_{10}r_9 + r_{10}$	$2156 = 2 \times 1078 + 0$	$d = \gcd(1078, 0) = 1078$	
Therefore, $d = \gcd(1160718174, 316258250) = 1078$			

The Extended Euclidean Algorithm

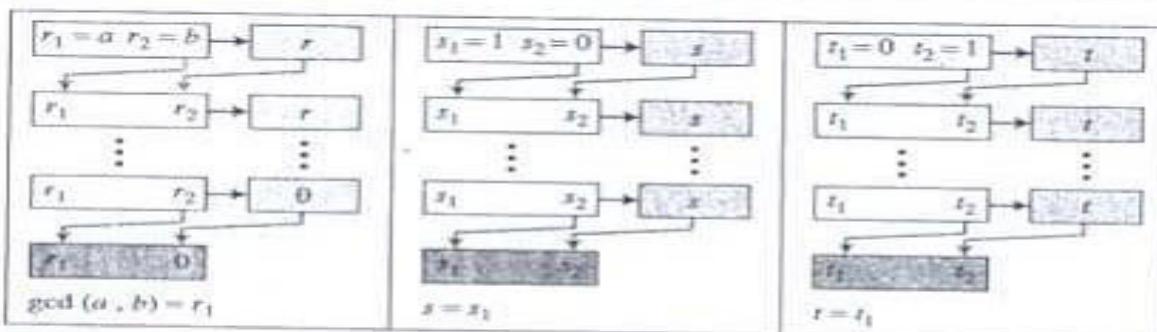
Given two integers a and b , we often need to find other two integers, s and t , such that

$$s \times a + t \times b = \gcd(a, b)$$

The extended Euclidean algorithm can calculate the $\gcd(a, b)$ and at the same time calculate the value of s and t . The algorithm and the process is shown in Figure 2.8.

As shown in Figure 2.8, the extended Euclidean algorithm uses the same number of steps as the Euclidean algorithm. However, in each step, we use three sets of calculations and exchanges instead of one. The algorithm uses three sets of variables, r 's, s 's, and t 's.

Figure 2.8 Extended Euclidean algorithm



a. Process

```

x1 ← a; x2 ← b; r1 ← a;
x1 ← 1; x2 ← 0; s1 ← 1;
t1 ← 0; t2 ← 1; t ← 1;      (Initialization)

while (x2 > 0)
{
    q ← x1 / x2;
    r ← x1 - q * x2;          (Updating r's)
    x1 ← x2; x2 ← r;
    s ← s1 - q * s2;          (Updating s's)
    s2 ← s1; s1 ← s;
    t ← t1 - q * t2;          (Updating t's)
    t2 ← t1; t1 ← t;
}
gcd (a , b) ← x1; s ← s1; t ← t1

```

b. Algorithm

Given $a = 161$ and $b = 28$, find $\gcd(a, b)$ and the values of s and t .

Solution

$$r = r_1 - q \times r_2 \quad s = s_1 - q \times s_2 \quad t = t_1 - q \times t_2$$

We use a table to follow the algorithm.

$$s = 1 - 5(0)$$

q	r_1	r_2	r	s_1	s_2	s	t_1	t_2	t
5	161	28	21	1	0	-1	0	1	-5
1	28	21	7	0	1	-1	1	-5	6
-3	21	7	0	1	-1	4	-5	6	-23
	7	0	-1	-1	4	6	-23		

We get $\gcd(161, 28) = 7$, $s = -1$ and $t = 6$. The answers can be tested because we have

$$(-1) \times 161 + 6 \times 28 = 7$$

Primality

Prime Numbers

- An integer $p > 1$ is prime number, if its divisors are ± 1 and $\pm p$
- Any non negative integer $a > 1$ can be factored in the form as $a = p_1^{a_1} \times p_2^{a_2} \times \cdots \times p_t^{a_t}$
- where $p_1 < p_2 < \dots < p_t$ are prime numbers and where each a_i is a positive integer.
- This is known as the fundamental theorem of arithmetic
- Examples: $91 = 7 \times 13$, $3600 = 24 \times 32 \times 5^2$

Miller-Rabin Algorithm

- also known as Rabin-Miller algorithm, or the Rabin-Miller test, or the Miller-Rabin test
- typically used to test a large number for primality

Two Properties of Prime Numbers

- If p is prime and a is a positive integer less than p , then $a^2 \bmod p = 1$, if and only if either $a \bmod p = 1$ or $a \bmod p = -1 \bmod p = p - 1$
- Let p be a prime number greater than 2. We can then write $p - 1 = 2^k q$ with $k > 0$, q odd

Algorithm

TEST (n)

1. Find integers k , q , with $k > 0$, q odd, so that $(n - 1) = 2^k q$;
2. Select a random integer a , $1 < a < n - 1$;
3. **if** $a^q \bmod n = 1$ **then** return("inconclusive");
4. **for** $j = 0$ **to** $k - 1$ **do**
5. **if** $a^{2^j q} \bmod n = n - 1$ **then** return("inconclusive");
6. **return**("composite");

Chinese Remainder Theorem

the CRT says it is possible to reconstruct integers in a certain range from their residues modulo a set of pairwise relatively prime moduli

Formula

$$M = \prod_{i=1}^k m_i$$

where the m_i are pairwise relatively prime; that is, $\gcd(m_i, m_j) = 1$ for $1 \leq i, j \leq k$, and $i \neq j$.

Relatively Prime

- Two integers are relatively prime, if their only common positive integer factor is 1
- Example: 8, 15 are relatively prime because positive divisors of 8 are 1, 2, 4, 8. Positive divisors of 15 are 1, 3, 5, 15. Common positive factor = 1

TWO ASSERTION OF CRT

1. The mapping of Equation (8.7) is a one-to-one correspondence (called a bijection) between Z_M and the Cartesian product $Z_{m_1} \times Z_{m_2} \times \cdots \times Z_{m_k}$. That is, for every integer A such that $0 \leq A \leq M$, there is a unique k -tuple (a_1, a_2, \dots, a_k) with $0 \leq a_i < m_i$ that represents it, and for every such k -tuple (a_1, a_2, \dots, a_k) , there is a unique integer A in Z_M .
2. Operations performed on the elements of Z_M can be equivalently performed on the corresponding k -tuples by performing the operation independently in each coordinate position in the appropriate system.

The following is an example of a set of equations with different moduli:

$$x \equiv 2 \pmod{3}$$

$$x \equiv 3 \pmod{5}$$

$$x \equiv 2 \pmod{7}$$

The solution to this set of equations is given in the next section; for the moment, note that the answer to this set of equations is $x = 23$. This value satisfies all equations: $23 \equiv 2 \pmod{3}$, $23 \equiv 3 \pmod{5}$, and $23 \equiv 2 \pmod{7}$.

Solution

The solution to the set of equations follows these steps:

1. Find $M = m_1 \times m_2 \times \dots \times m_k$. This is the common modulus.
2. Find $M_1 = M/m_1, M_2 = M/m_2, \dots, M_k = M/m_k$.
3. Find the multiplicative inverse of M_1, M_2, \dots, M_k using the corresponding moduli (m_1, m_2, \dots, m_k). Call the inverses $M_1^{-1}, M_2^{-1}, \dots, M_k^{-1}$.
4. The solution to the simultaneous equations is

$$x = (a_1 \times M_1 \times M_1^{-1} + a_2 \times M_2 \times M_2^{-1} + \dots + a_k \times M_k \times M_k^{-1}) \pmod{M}$$

Note that the set of equations can have a solution even if the moduli are not relatively prime but meet other conditions. However, in cryptography, we are only interested in solving equations with coprime moduli.

From the previous example, we already know that the answer is $x = 23$. We follow the four steps.

1. $M = 3 \times 5 \times 7 = 105$
2. $M_1 = 105 / 3 = 35, M_2 = 105 / 5 = 21, M_3 = 105 / 7 = 15$
3. The inverses are $M_1^{-1} = 2, M_2^{-1} = 1, M_3^{-1} = 1$
4. $x = (2 \times 35 \times 2 + 3 \times 21 \times 1 + 2 \times 15 \times 1) \pmod{105} = 23 \pmod{105}$

DISCRETE LOGARITHMS

More generally, we can say that the highest possible exponent to which a number can belong $(\bmod n)$ is $\phi(n)$. If a number is of this order, it is referred to as a **primitive root** of n . The importance of this notion is that if a is a primitive root of n , then its powers

$$a, a^2, \dots, a^{\phi(n)}$$

are distinct $(\bmod n)$ and are all relatively prime to n . In particular, for a prime number p , if a is a primitive root of p , then

$$a, a^2, \dots, a^{p-1}$$

are distinct $(\bmod p)$. For the prime number 19, its primitive roots are 2, 3, 10, 13, 14, and 15.

Logarithms for Modular Arithmetic

With ordinary positive real numbers, the logarithm function is the inverse of exponentiation. An analogous function exists for modular arithmetic.

Let us briefly review the properties of ordinary logarithms. The logarithm of a number is defined to be the power to which some positive base (except 1) must be raised in order to equal the number. That is, for base x and for a value y ,

$$y = x^{\log_x(y)}$$

The properties of logarithms include

$$\log_x(1) = 0$$

$$\log_x(x) = 1$$

$$\log_x(yz) = \log_x(y) + \log_x(z) \quad (8.11)$$

$$\log_x(y^r) = r \times \log_x(y) \quad (8.12)$$

Consider a primitive root a for some prime number p (the argument can be developed for nonprimes as well). Then we know that the powers of a from 1 through $(p - 1)$ produce each integer from 1 through $(p - 1)$ exactly once. We also know that any integer b satisfies

$$b = r \pmod{p} \quad \text{for some } r, \text{ where } 0 \leq r \leq (p - 1)$$

by the definition of modular arithmetic. It follows that for any integer b and a primitive root a of prime number p , we can find a unique exponent i such that

$$b = a^i \pmod{p} \quad \text{where } 0 \leq i \leq (p - 1)$$

This exponent i is referred to as the **discrete logarithm** of the number b for the base a $(\bmod p)$. We denote this value as $\text{dlog}_{a,p}(b)$.¹⁰

Note the following:

$$\text{dlog}_{a,p}(1) = 0 \quad \text{because } a^0 \pmod{p} = 1 \pmod{p} = 1 \quad (8.13)$$

$$\text{dlog}_{a,p}(a) = 1 \quad \text{because } a^1 \pmod{p} = a \quad (8.14)$$

Here is an example using a nonprime modulus, $n = 9$. Here $\phi(n) = 6$ and $a = 2$ is a primitive root. We compute the various powers of a and find

$$2^0 = 1 \quad 2^4 = 7 \pmod{9}$$

$$2^1 = 2 \quad 2^5 = 5 \pmod{9}$$

$$2^2 = 4 \quad 2^6 = 1 \pmod{9}$$

$$2^3 = 8$$

This gives us the following table of the numbers with given discrete logarithms $(\bmod 9)$ for the root $a = 2$:

Logarithm	0	1	2	3	4	5
Number	1	2	4	8	7	5

To make it easy to obtain the discrete logarithms of a given number, we rearrange the table:

Number	1	2	4	5	7	8
Logarithm	0	1	2	5	4	3

Now consider

$$x = a^{\text{dlog}_{a,p}(x)} \pmod{p} \quad y = a^{\text{dlog}_{a,p}(y)} \pmod{p}$$

$$xy = a^{\text{dlog}_{a,p}(xy)} \pmod{p}$$

Using the rules of modular multiplication,

$$\begin{aligned} xy \pmod{p} &= [(x \pmod{p})(y \pmod{p})] \pmod{p} \\ a^{\text{dlog}_{a,p}(xy)} \pmod{p} &= [(a^{\text{dlog}_{a,p}(x)} \pmod{p})(a^{\text{dlog}_{a,p}(y)} \pmod{p})] \pmod{p} \\ &= (a^{\text{dlog}_{a,p}(x) + \text{dlog}_{a,p}(y)}) \pmod{p} \end{aligned}$$

But now consider Euler's theorem, which states that, for every a and n that are relatively prime,

$$a^{\phi(n)} = 1 \pmod{n}$$

Any positive integer z can be expressed in the form $z = q + k\phi(n)$, with $0 \leq q < \phi(n)$. Therefore, by Euler's theorem,

$$a^z = a^q (\pmod{n}) \quad \text{if } z = q \pmod{\phi(n)}$$

Applying this to the foregoing equality, we have

$$\text{dlog}_{a,p}(xy) = [\text{dlog}_{a,p}(x) + \text{dlog}_{a,p}(y)] \pmod{\phi(p)}$$

and generalizing,

$$\text{dlog}_{a,p}(y^r) = [r \times \text{dlog}_{a,p}(y)] \pmod{\phi(p)}$$

This demonstrates the analogy between true logarithms and discrete logarithms.

Table 8.4 Tables of Discrete Logarithms, Modulo 19

(a) Discrete logarithms to the base 2, modulo 19

a	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
$\log_{2,19}(a)$	18	1	13	2	16	14	6	3	8	17	12	15	5	7	11	4	10	9

(b) Discrete logarithms to the base 3, modulo 19

a	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
$\log_{3,19}(a)$	18	7	1	14	4	8	6	3	2	11	12	15	17	13	5	10	16	9

(c) Discrete logarithms to the base 10, modulo 19

a	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
$\log_{10,19}(a)$	18	17	5	16	2	4	12	15	10	1	6	3	13	11	7	14	8	9

(d) Discrete logarithms to the base 13, modulo 19

a	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
$\log_{13,19}(a)$	18	11	17	4	14	10	12	15	16	7	6	3	1	5	13	8	2	9

(e) Discrete logarithms to the base 14, modulo 19

a	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
$\log_{14,19}(a)$	18	13	7	8	10	2	6	3	14	5	12	15	11	1	17	16	4	9

(f) Discrete logarithms to the base 15, modulo 19

a	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
$\log_{15,19}(a)$	18	5	11	10	8	16	12	15	4	13	6	3	7	17	1	2	14	9

Modular Arithmetic

The Modulus

If a is an integer and n is a positive integer, we define $a \bmod n$ to be the remainder when a is divided by n . The integer n is called the **modulus**. Thus, for any integer a , we can rewrite Equation (4.1) as follows:

$$a = qn + r \quad 0 \leq r < n; q = \lfloor a/n \rfloor$$

$$a = \lfloor a/n \rfloor \times n + (a \bmod n)$$

$$11 \bmod 7 = 4; \quad -11 \bmod 7 = 3$$

Two integers a and b are said to be **congruent modulo n** , if $(a \bmod n) = (b \bmod n)$. This is written as $a \equiv b \pmod{n}$.²

$$73 \equiv 4 \pmod{23}; \quad 21 \equiv -9 \pmod{10}$$

Note that if $a \equiv 0 \pmod{n}$, then $n|a$.

Properties of Congruences

1. $a \equiv b \pmod{n}$ if $n|(a - b)$.
2. $a \equiv b \pmod{n}$ implies $b \equiv a \pmod{n}$.
3. $a \equiv b \pmod{n}$ and $b \equiv c \pmod{n}$ imply $a \equiv c \pmod{n}$.

To demonstrate the first point, if $n|(a - b)$, then $(a - b) = kn$ for some k . So we can write $a = b + kn$. Therefore, $(a \bmod n) = (\text{remainder when } b + kn \text{ is divided by } n) = (\text{remainder when } b \text{ is divided by } n) = (b \bmod n)$.

$$\begin{aligned} 23 &\equiv 8 \pmod{5} & \text{because } 23 - 8 = 15 = 5 \times 3 \\ -11 &\equiv 5 \pmod{8} & \text{because } -11 - 5 = -16 = 8 \times (-2) \\ 81 &\equiv 0 \pmod{27} & \text{because } 81 - 0 = 81 = 27 \times 3 \end{aligned}$$

Modular Arithmetic Operations

1. $[(a \bmod n) + (b \bmod n)] \bmod n = (a + b) \bmod n$
2. $[(a \bmod n) - (b \bmod n)] \bmod n = (a - b) \bmod n$
3. $[(a \bmod n) \times (b \bmod n)] \bmod n = (a \times b) \bmod n$

We demonstrate the first property. Define $(a \bmod n) = r_a$ and $(b \bmod n) = r_b$. Then we can write $a = r_a + jn$ for some integer j and $b = r_b + kn$ for some integer k . Then

$$\begin{aligned} (a + b) \bmod n &= (r_a + jn + r_b + kn) \bmod n \\ &= (r_a + r_b + (k + j)n) \bmod n \\ &= (r_a + r_b) \bmod n \\ &= [(a \bmod n) + (b \bmod n)] \bmod n \end{aligned}$$

Examples

$$\begin{aligned}
 11 \bmod 8 &= 3; 15 \bmod 8 = 7 \\
 [(11 \bmod 8) + (15 \bmod 8)] \bmod 8 &= 10 \bmod 8 = 2 \\
 (11 + 15) \bmod 8 &= 26 \bmod 8 = 2 \\
 [(11 \bmod 8) - (15 \bmod 8)] \bmod 8 &= -4 \bmod 8 = 4 \\
 (11 - 15) \bmod 8 &= -4 \bmod 8 = 4 \\
 [(11 \bmod 8) \times (15 \bmod 8)] \bmod 8 &= 21 \bmod 8 = 5 \\
 (11 \times 15) \bmod 8 &= 165 \bmod 8 = 5
 \end{aligned}$$

To find $11^7 \bmod 13$, we can proceed as follows:

$$\begin{aligned}
 11^2 &= 121 = 4 \pmod{13} \\
 11^4 &= (11^2)^2 = 4^2 = 3 \pmod{13} \\
 11^7 &= 11 \times 4 \times 3 = 132 = 2 \pmod{13}
 \end{aligned}$$

Properties of Modular Arithmetic

set of residues, or residue classes ($\bmod n$)

Define the set Z_n as the set of nonnegative integers less than n :

$$Z_n = \{0, 1, \dots, (n - 1)\}$$

precise, each integer in Z_n represents a residue class. We can label the residue classes ($\bmod n$) as $[0], [1], [2], \dots, [n - 1]$, where

$$[r] = \{a: a \text{ is an integer, } a \equiv r \pmod{n}\}$$

The residue classes ($\bmod 4$) are

$$\begin{aligned}
 [0] &= \{\dots, -16, -12, -8, -4, 0, 4, 8, 12, 16, \dots\} \\
 [1] &= \{\dots, -15, -11, -7, -3, 1, 5, 9, 13, 17, \dots\} \\
 [2] &= \{\dots, -14, -10, -6, -2, 2, 6, 10, 14, 18, \dots\} \\
 [3] &= \{\dots, -13, -9, -5, -1, 3, 7, 11, 15, 19, \dots\}
 \end{aligned}$$

reducing k modulo n .

Finding the smallest nonnegative integer to which k is congruent modulo n

if $(a + b) \equiv (a + c) \pmod{n}$ **then** $b \equiv c \pmod{n}$

$$(5 + 23) \equiv (5 + 7) \pmod{8}; 23 \equiv 7 \pmod{8}$$

if $(a \times b) \equiv (a \times c) \pmod{n}$ **then** $b \equiv c \pmod{n}$ if a is relatively prime to n

Properties of Modular Arithmetic for Integers in Z_n

Commutative Laws	$(w + x) \bmod n = (x + w) \bmod n$ $(w \times x) \bmod n = (x \times w) \bmod n$
Associative Laws	$[(w + x) + y] \bmod n = [w + (x + y)] \bmod n$ $[(w \times x) \times y] \bmod n = [w \times (x \times y)] \bmod n$
Distributive Law	$[w \times (x + y)] \bmod n = [(w \times x) + (w \times y)] \bmod n$
Identities	$(0 + w) \bmod n = w \bmod n$ $(1 \times w) \bmod n = w \bmod n$
Additive Inverse ($-w$)	For each $w \in Z_n$, there exists a z such that $w + z \equiv 0 \pmod{n}$

Table 4.2 Arithmetic Modulo 8

+	0	1	2	3	4	5	6	7
0	0	1	2	3	4	5	6	7
1	1	2	3	4	5	6	7	0
2	2	3	4	5	6	7	0	1
3	3	4	5	6	7	0	1	2
4	4	5	6	7	0	1	2	3
5	5	6	7	0	1	2	3	4
6	6	7	0	1	2	3	4	5
7	7	0	1	2	3	4	5	6

(a) Addition modulo 8

*	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7
2	0	2	4	6	0	2	4	6
3	0	3	6	1	4	7	2	5
4	0	4	0	4	0	4	0	4
5	0	5	2	7	4	1	6	3
6	0	6	4	2	0	6	4	2
7	0	7	6	5	4	3	2	1

(b) Multiplication modulo 8

*	0	1	2	3
0	0	0	—	—
1	0	1	1	—
2	0	2	—	—
3	0	3	3	—
4	0	4	—	—
5	0	5	5	—
6	0	6	—	—
7	0	7	1	7

(c) Additive and multiplicative inverses modulo 8

GROUPS, RINGS AND FIELDS

Groups, rings, and fields are the fundamental elements of a branch of mathematics known as abstract algebra, or modern algebra. In abstract algebra, we are concerned with sets on whose elements we can operate algebraically; that is, we can combine two elements of the set, perhaps in several ways, to obtain a third element of the set. These operations are subject to specific rules, which define the nature of the set. By convention, the notation for the two principal classes of operations on set elements is usually the same as the notation for addition and multiplication on ordinary numbers. However, it is important to note that, in abstract algebra, we are not limited to ordinary arithmetical operations. All this should become clear as we proceed.

Groups

A **group** G , sometimes denoted by $\{G, \cdot\}$, is a set of elements with a binary operation denoted by \cdot that associates to each ordered pair (a, b) of elements in G an element $(a \cdot b)$ in G , such that the following axioms are obeyed:⁴

- | | |
|-------------------------------|---|
| (A1) Closure: | If a and b belong to G , then $a \cdot b$ is also in G . |
| (A2) Associative: | $a \cdot (b \cdot c) = (a \cdot b) \cdot c$ for all a, b, c in G . |
| (A3) Identity element: | There is an element e in G such that $a \cdot e = e \cdot a = a$ for all a in G . |
| (A4) Inverse element: | For each a in G , there is an element a' in G such that $a \cdot a' = a' \cdot a = e$. |

If a group has a finite number of elements, it is referred to as a **finite group**, and the **order of the group** is equal to the number of elements in the group. Otherwise, the group is an **infinite group**.

A group is said to be **abelian** if it satisfies the following additional condition:

- (A5) Commutative:** $a \cdot b = b \cdot a$ for all a, b in G .

The set of integers (positive, negative, and 0) under addition is an abelian group. The set of nonzero real numbers under multiplication is an abelian group. The set S_n from the preceding example is a group but not an abelian group for $n > 2$.

When the group operation is addition, the identity element is 0; the inverse element of a is $-a$; and subtraction is defined with the following rule: $a - b = a + (-b)$.

CYCLIC GROUP We define exponentiation within a group as a repeated application of the group operator, so that $a^3 = a \cdot a \cdot a$. Furthermore, we define $a^0 = e$ as the identity element, and $a^{-n} = (a')^n$, where a' is the inverse element of a within the group. A group G is cyclic if every element of G is a power a^k (k is an integer) of a fixed element $a \in G$. The element a is said to generate the group G or to be a **generator** of G . A cyclic group is always abelian and may be finite or infinite.

The additive group of integers is an infinite cyclic group generated by the element 1. In this case, powers are interpreted additively, so that n is the n th power of 1.

Rings

A **ring** R , sometimes denoted by $\{R, +, \times\}$, is a set of elements with two binary operations, called **addition** and **multiplication**,⁵ such that for all a, b, c in R the following axioms are obeyed.

- (A1–A5)** R is an abelian group with respect to addition; that is, R satisfies axioms A1 through A5. For the case of an additive group, we denote the identity element as 0 and the inverse of a as $-a$.

- | | |
|--|--|
| (M1) Closure under multiplication: | If a and b belong to R , then ab is also in R . |
| (M2) Associativity of multiplication: | $a(bc) = (ab)c$ for all a, b, c in R . |
| (M3) Distributive laws: | $a(b + c) = ab + ac$ for all a, b, c in R .
$(a + b)c = ac + bc$ for all a, b, c in R . |

In essence, a ring is a set in which we can do addition, subtraction [$a - b = a + (-b)$], and multiplication without leaving the set.

With respect to addition and multiplication, the set of all n -square matrices over the real numbers is a ring.

A ring is said to be commutative if it satisfies the following additional condition:

(M4) Commutativity of multiplication: $ab = ba$ for all a, b in R .

Let S be the set of even integers (positive, negative, and 0) under the usual operations of addition and multiplication. S is a commutative ring. The set of all n -square matrices defined in the preceding example is not a commutative ring.

The set Z_n of integers $\{0, 1, \dots, n - 1\}$, together with the arithmetic operations modulo n , is a commutative ring (Table 4.3).

Next, we define an integral domain, which is a commutative ring that obeys the following axioms.

(M5) Multiplicative identity: There is an element 1 in R such that $a1 = 1a = a$ for all a in R .

(M6) No zero divisors: If a, b in R and $ab = 0$, then either $a = 0$ or $b = 0$.

Let S be the set of integers, positive, negative, and 0, under the usual operations of addition and multiplication. S is an integral domain.

Fields

A field F , sometimes denoted by $\{F, +, \times\}$, is a set of elements with two binary operations, called *addition* and *multiplication*, such that for all a, b, c in F the following axioms are obeyed.

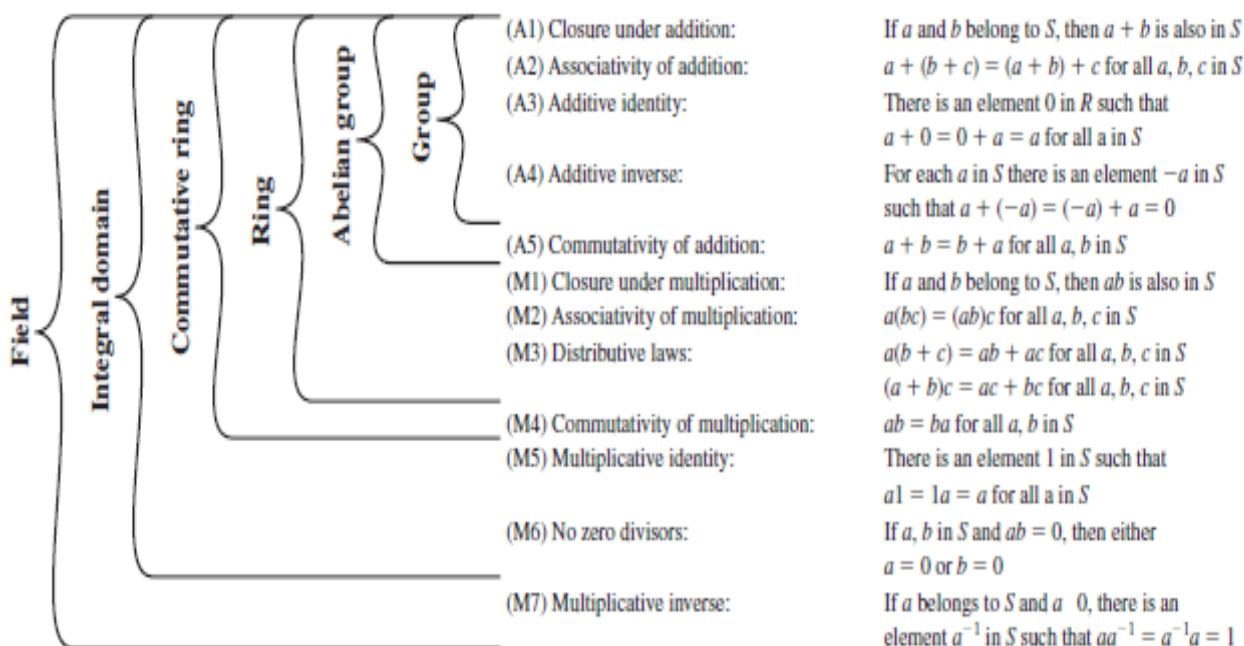
(A1–M6) F is an integral domain; that is, F satisfies axioms A1 through A5 and M1 through M6.

(M7) Multiplicative inverse: For each a in F , except 0, there is an element a^{-1} in F such that $aa^{-1} = (a^{-1})a = 1$.

In essence, a field is a set in which we can do addition, subtraction, multiplication, and division without leaving the set. Division is defined with the following rule: $a/b = a(b^{-1})$.

Familiar examples of fields are the rational numbers, the real numbers, and the complex numbers. Note that the set of all integers is not a field, because not every element of the set has a multiplicative inverse; in fact, only the elements 1 and -1 have multiplicative inverses in the integers.

Figure 4.2 summarizes the axioms that define groups, rings, and fields.



FINITE FIELDS OF THE FORM GF(p^n)

The finite field of order p^n is generally written GF(p^n); GF stands for Galois field, in honor of the mathematician who first studied finite fields. Two special cases are of interest for our purposes. For $n = 1$, we have the finite field GF(p); this finite field has a different structure than that for finite fields with $n > 1$ and is studied in this section. In Section 4.7, we look at finite fields of the form GF(2^n).

Finite Fields of Order p

For a given prime, p , we define the finite field of order p , GF(p), as the set Z_p of integers $\{0, 1, \dots, p - 1\}$ together with the arithmetic operations modulo p .

Recall that we showed in Section 4.3 that the set Z_n of integers $\{0, 1, \dots, n - 1\}$, together with the arithmetic operations modulo n , is a commutative ring (Table 4.3). We further observed that any integer in Z_n has a multiplicative inverse if and only if that integer is relatively prime to n [see discussion of Equation (4.5)].⁷ If n is prime, then all of the nonzero integers in Z_n are relatively prime to n , and therefore there exists a multiplicative inverse for all of the nonzero integers in Z_n . Thus, for Z_p we can add the following properties to those listed in Table 4.3:

Multiplicative inverse (w^{-1})	For each $w \in Z_p$, $w \neq 0$, there exists a $z \in Z_p$ such that $w \times z = 1 \pmod{p}$
-------------------------------------	--

The simplest finite field is GF(2). Its arithmetic operations are easily summarized:

+	0	1
0	0	1
1	1	0

Addition

×	0	1
0	0	0
1	0	1

Multiplication

w	$-w$	w^{-1}
0	0	—
1	1	1

Inverses

In this case, addition is equivalent to the exclusive-OR (XOR) operation, and multiplication is equivalent to the logical AND operation.

Finding the Multiplicative Inverse in GF(p)

It is easy to find the multiplicative inverse of an element in GF(p) for small values of p . You simply construct a multiplication table, such as shown in Table 4.5b, and the desired result can be read directly. However, for large values of p , this approach is not practical.

Table 4.5 Arithmetic in GF(7)

+	0	1	2	3	4	5	6
0	0	1	2	3	4	5	6
1	1	2	3	4	5	6	0
2	2	3	4	5	6	0	1
3	3	4	5	6	0	1	2
4	4	5	6	0	1	2	3
5	5	6	0	1	2	3	4
6	6	0	1	2	3	4	5

(a) Addition modulo 7

×	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6
2	0	2	4	6	1	3	5
3	0	3	6	2	5	1	4
4	0	4	1	5	2	6	3
5	0	5	3	1	6	4	2
6	0	6	5	4	3	2	1

(b) Multiplication modulo 7

w	$-w$	w^{-1}
0	0	—
1	6	1
2	5	4
3	4	5
4	3	2
5	2	3
6	1	6

(c) Additive and multiplicative inverses modulo 7

POLYNOMIAL ARITHMETIC

Three classes of polynomial arithmetic:

- Ordinary polynomial arithmetic, using the basic rules of algebra.
- Polynomial arithmetic in which the arithmetic on the coefficients is performed modulo p ; that is, the coefficients are in GF(p).
- Polynomial arithmetic in which the coefficients are in GF(p), and the polynomials are defined modulo a polynomial $m(x)$ whose highest power is some integer n .

$\begin{array}{r} x^3 + x^2 + 2 \\ + (x^2 - x + 1) \\ \hline x^3 + 2x^2 - x + 3 \end{array}$	$\begin{array}{r} x^3 + x^2 + 2 \\ - (x^2 - x + 1) \\ \hline x^3 + x + 1 \end{array}$
(a) Addition	(b) Subtraction

$\begin{array}{r} x^3 + x^2 + 2 \\ \times (x^2 - x + 1) \\ \hline x^3 + x^2 + 2 \\ - x^4 - x^3 - 2x \\ \hline x^5 + x^4 + 2x^2 \\ \hline x^5 + 3x^4 - 2x + 2 \end{array}$	$\begin{array}{r} x + 2 \\ \hline x^2 - x + 1 \sqrt{x^3 + x^2 + 2} \\ \hline x^3 - x^2 + x \\ \hline 2x^2 - x + 2 \\ \hline 2x^2 - 2x + 2 \\ \hline x \end{array}$
(c) Multiplication	(d) Division

Figure 4.3 Examples of Polynomial Arithmetic

A polynomial $f(x)$ over a field F is called **irreducible** if and only if $f(x)$ cannot be expressed as a product of two polynomials, both over F , and both of degree lower than that of $f(x)$. By analogy to integers, an irreducible polynomial is also called a **prime polynomial**.

The polynomial⁹ $f(x) = x^4 + 1$ over $\text{GF}(2)$ is reducible, because $x^4 + 1 = (x + 1)(x^3 + x^2 + x + 1)$.

Consider the polynomial $f(x) = x^3 + x + 1$. It is clear by inspection that x is not a factor of $f(x)$. We easily show that $x + 1$ is not a factor of $f(x)$:

$$\begin{array}{r} x^2 + x \\ x + 1 \sqrt{x^3 + x + 1} \\ \hline x^3 + x^2 \\ x^2 + x \\ \hline x^2 + x \\ \hline 1 \end{array}$$

Thus, $f(x)$ has no factors of degree 1. But it is clear by inspection that if $f(x)$ is reducible, it must have one factor of degree 2 and one factor of degree 1. Therefore, $f(x)$ is irreducible.

$\begin{array}{r} x^7 + x^5 + x^4 + x^3 + x + 1 \\ + (x^3 + x + 1) \\ \hline x^7 + x^5 + x^4 \end{array}$	$\begin{array}{r} x^7 + x^5 + x^4 + x^3 + x + 1 \\ - (x^3 + x + 1) \\ \hline x^7 + x^5 + x^4 \end{array}$
(a) Addition	(b) Subtraction

$\begin{array}{r} x^7 + x^5 + x^4 + x^3 + x + 1 \\ \times (x^3 + x + 1) \\ \hline x^7 + x^5 + x^4 + x^3 + x^2 + x \\ x^{10} + x^8 + x^7 + x^6 + x^5 + x^4 + x^3 + x^2 + x \\ \hline x^{10} + x^4 + x^2 + 1 \end{array}$
(c) Multiplication

$\begin{array}{r} x^4 + 1 \\ \hline x^3 + x + 1 \sqrt{x^7 + x^5 + x^4 + x^3 + x + 1} \\ \hline x^7 + x^5 + x^4 \\ \hline x^3 + x + 1 \end{array}$
(d) Division

Figure 4.4 Examples of Polynomial Arithmetic over $\text{GF}(2)$

FINITE FIELDS OF THE FORM GF(2ⁿ)

Table 4.6 Arithmetic in GF(2³)

	000	001	010	011	100	101	110	111
+	0	1	2	3	4	5	6	7
000	0	1	2	3	4	5	6	7
001	1	0	3	2	5	4	7	6
010	2	3	0	1	6	7	4	5
011	3	2	1	0	7	6	5	4
100	4	5	6	7	0	1	2	3
101	5	4	7	6	1	0	3	2
110	6	7	4	5	2	3	0	1
111	7	6	5	4	3	2	1	0

(a) Addition

	000	001	010	011	100	101	110	111
×	0	1	2	3	4	5	6	7
000	0	0	0	0	0	0	0	0
001	1	0	1	2	3	4	5	6
010	2	0	2	4	6	3	1	7
011	3	0	3	6	5	7	4	2
100	4	0	4	3	7	6	2	5
101	5	0	5	1	4	2	7	3
110	6	0	6	7	1	5	3	4
111	7	0	7	5	2	1	6	3

(b) Multiplication

w	-w	w ⁻¹
0	0	—
1	1	1
2	2	5
3	3	6
4	4	7
5	5	2
6	6	3
7	7	4

(c) Additive and multiplicative inverses

Table 4.7 Polynomial Arithmetic Modulo ($x^3 + x + 1$)

	000	001	010	011	100	101	110	111
+	0	1	x	$x + 1$	x^2	$x^2 + 1$	$x^2 + x$	$x^2 + x + 1$
000	0	0	1	x	$x + 1$	x^2	$x^2 + 1$	$x^2 + x + 1$
001	1	1	0	$x + 1$	x	$x^2 + 1$	x^2	$x^2 + x + 1$
010	x	x	$x + 1$	0	1	$x^2 + x$	x^2	$x^2 + 1$
011	$x + 1$	$x + 1$	x	1	0	$x^2 + x + 1$	$x^2 + x$	x^2
100	x^2	$x^2 + 1$	$x^2 + x$	$x^2 + x + 1$	0	1	x	$x + 1$
101	$x^2 + 1$	$x^2 + 1$	$x^2 + x + 1$	$x^2 + x$	1	0	$x + 1$	x
110	$x^2 + x$	$x^2 + x + 1$	x^2	$x^2 + 1$	x	$x + 1$	0	1
111	$x^2 + x + 1$	$x^2 + x$	$x^2 + 1$	x^2	$x + 1$	x	1	0

(a) Addition

	000	001	010	011	100	101	110	111
×	0	1	x	$x + 1$	x^2	$x^2 + 1$	$x^2 + x$	$x^2 + x + 1$
000	0	0	0	0	0	0	0	0
001	0	1	x	$x + 1$	x^2	$x^2 + 1$	$x^2 + x$	$x^2 + x + 1$
010	0	x	x^2	$x^2 + x$	$x + 1$	1	$x^2 + x + 1$	$x^2 + 1$
011	0	$x + 1$	$x^2 + x$	$x^2 + 1$	$x^2 + x + 1$	x^2	1	x
100	0	x^2	$x + 1$	$x^2 + x + 1$	$x^2 + x$	x	$x^2 + 1$	1
101	0	$x^2 + 1$	1	x^2	x	$x^2 + x + 1$	$x + 1$	$x^2 + x$
110	0	$x^2 + x$	$x^2 + x + 1$	1	$x^2 + 1$	$x + 1$	x	x^2
111	0	$x^2 + x + 1$	$x^2 + 1$	x	1	$x^2 + 1$	x^2	$x + 1$

(b) Multiplication

Fermat & Euler Theorem

- play important roles in public-key cryptography

Fermat's Theorem

If p is prime and a is a positive integer not divisible by p , then

$$a^{p-1} \equiv 1 \pmod{p}$$

Proof:

- Consider the set of positive integers less than p : $\{1, 2, 3, \dots, p-1\}$
- Multiply each element by a , modulo p to get the set
 - $X = \{a \pmod{p}, 2a \pmod{p}, \dots, (p-1)a \pmod{p}\}$
- None of the elements of X is equal to zero because p does not divide a
- Therefore, we know that the $(p-1)$ elements of X are all positive integers with no two elements equal
- We can conclude the X consists of the set of integers $\{1, 2, \dots, p-1\}$ in some order
- Multiplying the numbers in both sets (p and X) and taking the result mod p yields

$$\begin{aligned} a \times 2a \times \dots \times (p-1)a &= [(1 \times 2 \times \dots \times (p-1)) \pmod{p}] \\ a^{p-1}(p-1)! &\equiv (p-1)! \pmod{p} \end{aligned}$$

We can cancel the $(p-1)!$ term because it is relatively prime to p , which proves the theorem

Example:

$$\begin{aligned} a &= 7, p = 19 \\ 7^2 &\equiv 49 \equiv 11 \pmod{19} \\ 7^4 &\equiv 121 \equiv 7 \pmod{19} \\ 7^8 &\equiv 49 \equiv 11 \pmod{19} \\ 7^{16} &\equiv 121 \equiv 7 \pmod{19} \\ a^{p-1} &= 7^{18} = 7^{16} \times 7^2 = 7 \times 11 = 1 \pmod{19} \end{aligned}$$

An alternative form of Fermat's theorem

If p is prime and a is a positive integer, then $a^p \equiv a \pmod{p}$

$$\begin{aligned} p &= 5, a = 3 & a^p &= 3^5 = 243 \equiv 3 \pmod{5} = a \pmod{p} \\ p &= 5, a = 10 & a^p &= 10^5 = 100000 \equiv 10 \pmod{5} = a \pmod{p} \end{aligned}$$

Euler's Theorem

Euler's totient function ($\phi(n)$)

- the number of positive integers less than n and relatively prime to n . $\phi(1) = 1$
- for a prime number p , $\phi(p) = p - 1$
- for two prime numbers where $p \neq q$,

$$\phi(n) = \phi(pq) = \phi(p) \times \phi(q) = (p-1) \times (q-1)$$

$$\begin{aligned} \phi(21) &= \phi(3) \times \phi(7) = (3-1) \times (7-1) = 2 \times 6 = 12 \\ \text{where the } 12 \text{ integers are } &\{1, 2, 4, 5, 8, 10, 11, 13, 16, 17, 19, 20\}. \end{aligned}$$

To see that $\phi(n) = \phi(p) \times \phi(q)$, consider that the set of positive integers less than n is the set $\{1, \dots, (pq - 1)\}$. The integers in this set that are not relatively prime to n are the set $\{p, 2p, \dots, (q - 1)p\}$ and the set $\{q, 2q, \dots, (p - 1)q\}$. Accordingly,

$$\begin{aligned}\phi(n) &= (pq - 1) - [(q - 1) + (p - 1)] \\ &= pq - (p + q) + 1 \\ &= (p - 1) \times (q - 1) \\ &= \phi(p) \times \phi(q)\end{aligned}$$

Euler's Theorem

- for every a and n that are relatively prime $a^{\phi(n)} \equiv 1 \pmod{n}$
- alternative form $a^{\phi(n)+1} \equiv a \pmod{n}$

Proof:

Consider the set of positive integers less than n that are relatively prime to n , labeled as

$$R = [x_1, x_2, \dots, x_{\phi(n)}]$$

That is, each element x_i of R is a unique positive integer less than n with $\gcd(x_i, n) = 1$. Now multiply each element by a , modulo n :

$$S = \{(ax_1 \pmod{n}, ax_2 \pmod{n}, \dots, ax_{\phi(n)} \pmod{n})\}$$

The set S is a permutation⁶ of R , by the following line of reasoning:

- Because a is relatively prime to n and x_i is relatively prime to n , ax_i must also be relatively prime to n . Thus, all the members of S are integers that are less than n and that are relatively prime to n .
- There are no duplicates in S . Refer to Equation (4.5). If $ax_i \pmod{n} = ax_j \pmod{n}$, then $x_i = x_j$.

Therefore,

$$\begin{aligned}\prod_{i=1}^{\phi(n)} (ax_i \pmod{n}) &= \prod_{i=1}^{\phi(n)} x_i \\ \prod_{i=1}^{\phi(n)} ax_i &\equiv \prod_{i=1}^{\phi(n)} x_i \pmod{n} \\ a^{\phi(n)} \times \left[\prod_{i=1}^{\phi(n)} x_i \right] &= \prod_{i=1}^{\phi(n)} x_i \pmod{n} \\ a^{\phi(n)} &\equiv 1 \pmod{n}\end{aligned}$$

which completes the proof. This is the same line of reasoning applied to the proof of Fermat's theorem.

CS6701-

**CRYPTOGRAPHY AND NETWORK
SECURITY**

UNIT 2 NOTES

PREPARED BY
R.CYNTHIA PRIYADHARSHINI
AP/IT/SREC

Block Ciphers

- A block cipher is an encryption/decryption scheme in which **a block of plaintext is treated as a whole** and used to produce a ciphertext block of equal length.
- A **stream cipher** is one that encrypts a digital data stream **one bit or one byte at a time**.
 - Examples of classical stream ciphers are the autokeyed Vigenère cipher and the Vernam cipher.
- A **block cipher** is one in which a **block of plaintext** is treated as a whole and used to produce a ciphertext block of equal length
- Many block ciphers have a **Feistel structure**. Such a structure consists of a number of identical rounds of processing. In each round, a substitution is performed on one half of the data being processed, followed by a permutation that interchanges the two halves. The original key is expanded so that a different key is used for each round.
- The **Data Encryption Standard (DES)** has been the most widely used encryption algorithm until recently. It exhibits the classic Feistel structure. DES uses a 64-bit block and a 56-bit key.
- Two important methods of cryptanalysis are **differential cryptanalysis** and **linear cryptanalysis**. DES has been shown to be highly resistant to these two types of attack

Diffusion and Confusion

Shannon suggests two methods for frustrating statistical cryptanalysis: diffusion and confusion.

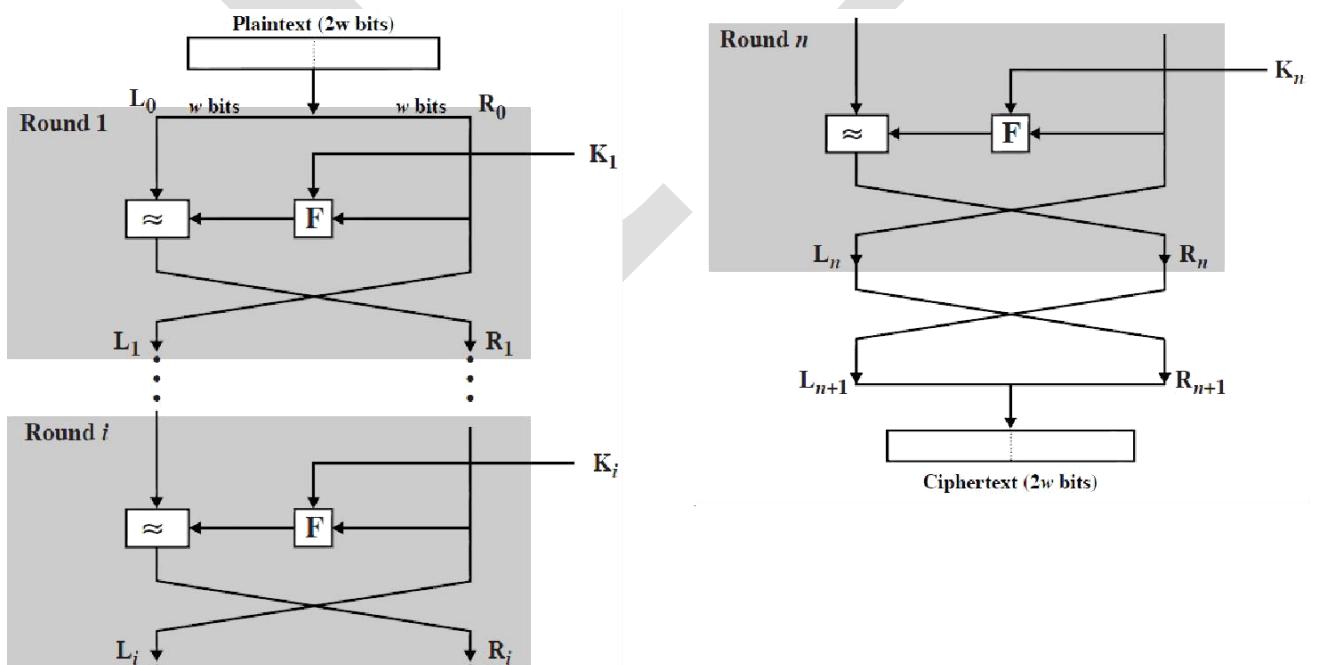
Diffusion

- the statistical structure of the plaintext is dissipated into long-range statistics of the ciphertext.
- This is achieved by having each plaintext digit affect the value of many ciphertext digits;
- generally this is equivalent to having each ciphertext digit be affected by many plaintext digits

Confusion

- seeks to make the relationship between the statistics of the ciphertext and the value of the encryption key as complex as possible, again to thwart attempts to discover the key.
- Thus, even if the attacker can get some handle on the statistics of the ciphertext, the way in which the key was used to produce that ciphertext is so complex as to make it difficult to deduce the key.

Feistel Cipher Structure



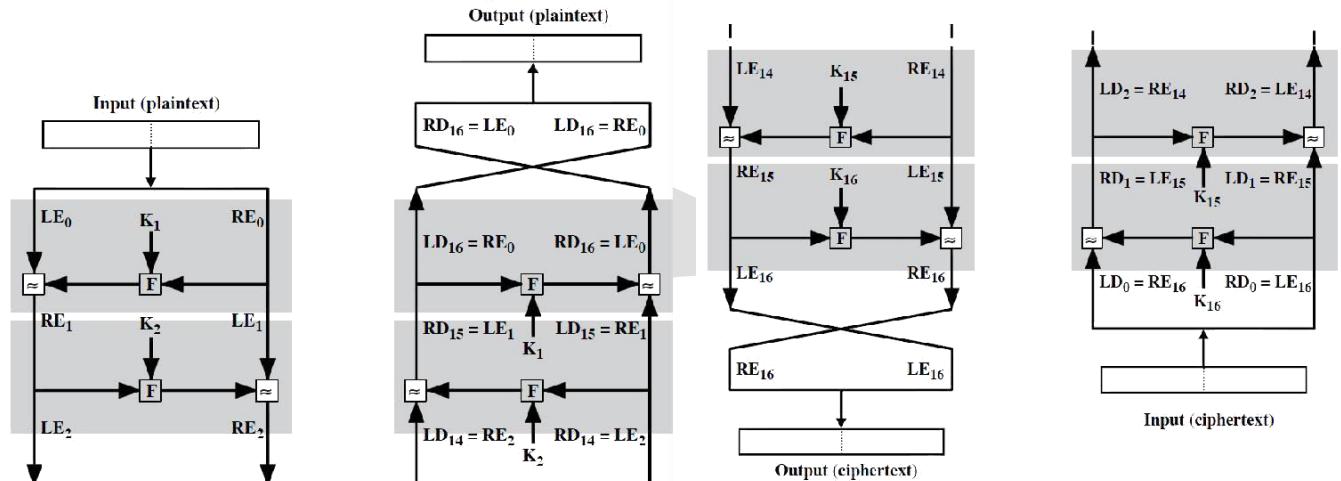
- All rounds have the same structure.
- A substitution is performed on the left half of the data.
- This is done by applying a round function F to the right half of the data and then taking the exclusive-OR of the output of that function and the left half of the data.
- The round function has the same general structure and parameterized by the round subkey K_{*i*}
- Following this, a permutation is performed that consists of the interchange of the two halves of the data

- This structure is a particular form of the substitution-permutation network (SPN)

Feistel network depends on the choice of the following parameters and design features

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

Feistel Encryption and Decryption



Simplified DES

- educational rather than a secure encryption algorithm.
- It has similar properties and structure to DES with much smaller parameters

Simplified DES Scheme

- The S-DES encryption algorithm takes an 8-bit block of plaintext (example: 10111101) and a 10-bit key as input and produces an 8-bit block of ciphertext as output.
- The S-DES decryption algorithm takes an 8-bit block of ciphertext and the same 10-bit key used to produce that ciphertext as input and produces the original 8-bit block of plaintext.

Involves five functions:

- an initial permutation (IP);
- a complex function labeled f_K , which involves both permutation and substitution operations and depends on a key input;
- a simple permutation function that switches (SW) the two halves of the data;
- the function f_K again;
- finally a permutation function that is the inverse of the initial permutation (IP^{-1}).

Algorithm

$$IP^{-1} \circ f_{K_2} \circ SW \circ f_{K_1} \circ IP$$

$$K_1 = P8(Shift(P10(key)))$$

$$\text{ciphertext} = IP^{-1}(f_{K_2}(SW(f_{K_1}(IP(\text{plaintext})))))$$

$$K_2 = P8(Shift(Shift(P10(key))))$$

$$\text{plaintext} = IP^{-1}(f_{K_1}(SW(f_{K_2}(IP(\text{ciphertext})))))$$

$$f_K(L, R) = (L \oplus F(R, SK), R)$$

P10							
3	5	2	7	4	10	1	9

P8							
6	3	7	4	8	5	10	9

IP							
2	6	3	1	4	8	5	7

P4			
2	4	3	1

IP ⁻¹							
4	1	3	5	7	2	8	6

E/P							
4	1	2	3	2	3	4	1

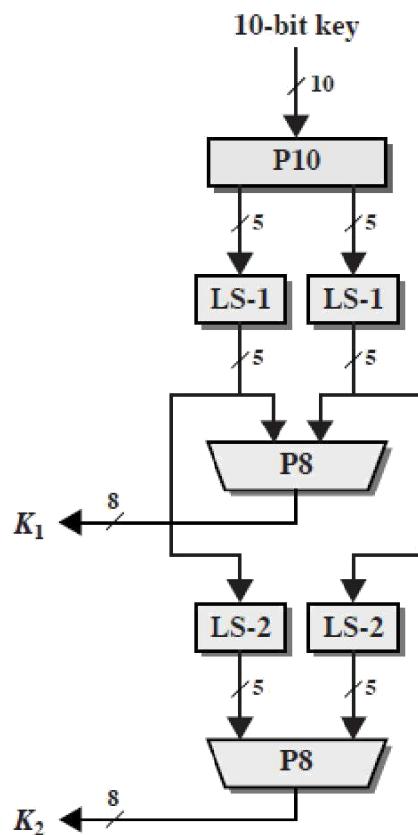
n_4	n_1	n_2	n_3
n_2	n_3	n_4	n_1

$$\begin{array}{c|c|c|c} n_4 \oplus k_{11} & n_1 \oplus k_{12} & n_2 \oplus k_{13} & n_3 \oplus k_{14} \\ n_2 \oplus k_{15} & n_3 \oplus k_{16} & n_4 \oplus k_{17} & n_1 \oplus k_{18} \end{array}$$

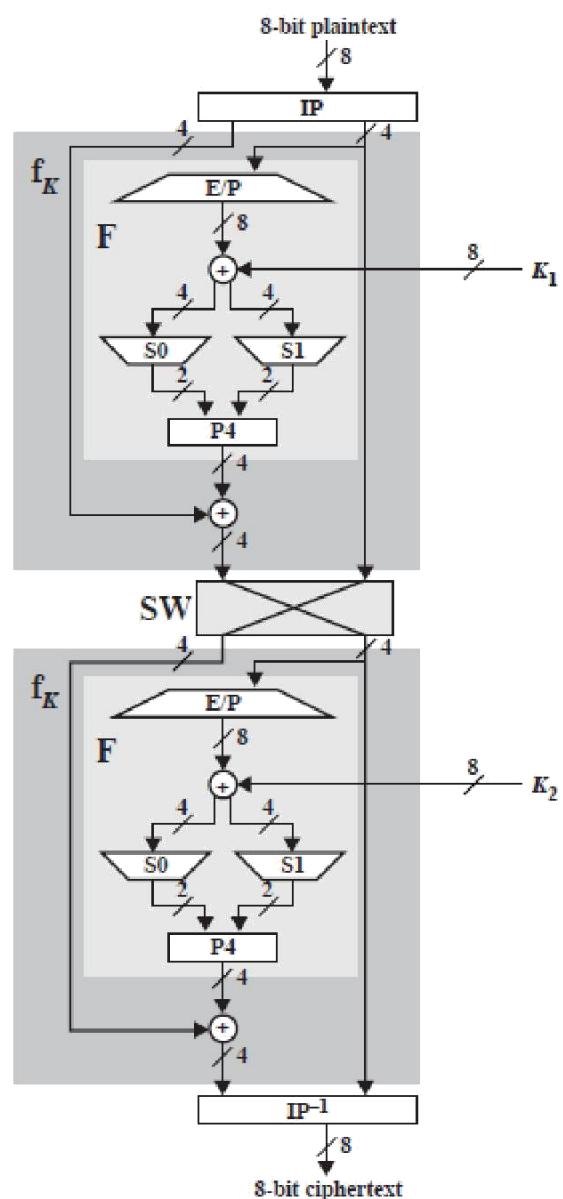
The 8-bit subkey $K_1 = (k_{11}, k_{12}, k_{13}, k_{14}, k_{15}, k_{16}, k_{17}, k_{18})$ is added
rename these 8 bits

$P_{0,0}$	$P_{0,1}$	$P_{0,2}$	$P_{0,3}$
$P_{1,0}$	$P_{1,1}$	$P_{1,2}$	$P_{1,3}$
	0 1 2 3		0 1 2 3
$S0 = 1$	$\begin{bmatrix} 1 & 0 & 3 & 2 \\ 3 & 2 & 1 & 0 \\ 2 & 0 & 2 & 1 \\ 3 & 1 & 3 & 2 \end{bmatrix}$	$S1 = 1$	$\begin{bmatrix} 0 & 1 & 2 & 3 \\ 2 & 0 & 1 & 3 \\ 2 & 3 & 0 & 1 \\ 2 & 1 & 0 & 3 \end{bmatrix}$

Key Generation for Simplified DES



Simplified DES Encryption Detail



Plain Text	1	2	3	4	5	6	7	8	9	10
Key	0	0	1	0	1	0	0	0	0	0
P10	1	1	0	0	0	1	1	1	1	0
LS-1	0	0	1	1	0	0	1	1	1	1
P8 (K1)	0	1	1	0	0	1	1	1	0	1
	6	3	7	4	8	5	10	9	1	0

LS-2	1 0 0 0 1	1 1 0 1 1
P8 (K2)	1 0 1 0 0	1 1 1
IP (PT)	2 6 3 1 4 8 5 7	0 0 1 0 0 0 1 0
R (IP)	0 0 1 0	
EP	4 1 2 3 2 3 4 1	0 0 0 1 0 1 0 0
K1	1 1 1 0	1 0 0 1
XOR	1 1 1 1	1 1 0 1
S0 (10 11)	S0 = 10	S1 = 0 0
P4	0 0 0 1	
P4 XOR L	0 0 1 1	
SWITCH	0 0 1 1 0 0 1 0	
	0 0 1 0 0 0 1 1	
EP(SW(R))	1 0 0 1 0 1 1 0	
XOR K2	1 0 1 0 0 1 1 1	
	0 0 1 1 0 0 0 1	
S0 & S1	1 0 1 0	
P4	0 0 1 1	
XOR L	0 0 1 0 0 0 1	
IP-1	0 0 0 1 0 0 1 1	
CT	1 0 0 0 1 0 1 0	

The Data Encryption Standard (DES)

- Overview
- DES Encryption
 - General Depiction of DES Encryption Algorithm
 - Initial Permutation
 - Permutation Tables for DES
 - Details of Single Round
 - Calculation of F(R, K)
 - Definition of DES S-Boxes
 - Key Generation
- DES Decryption
- The Avalanche Effect

Overview

- data are encrypted in 64-bit blocks using a 56-bit key.
- The algorithm transforms 64-bit input in a series of steps into a 64-bit output.
- The same steps, with the same key, are used to reverse the encryption

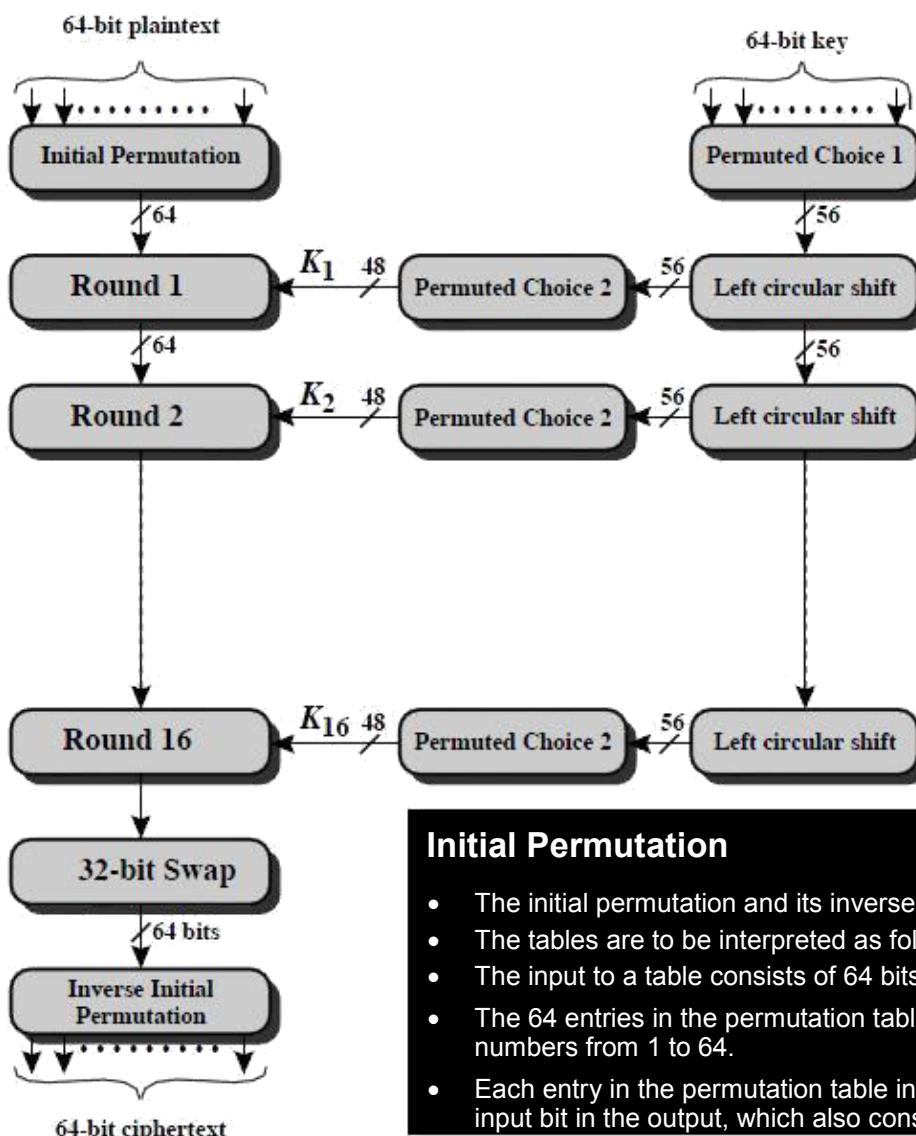
DES Encryption

- there are two inputs to the encryption function: the plaintext to be encrypted and the key.
- In this case, the plaintext must be 64 bits in length and the key is 56 bits in length

General Depiction of DES Encryption Algorithm

- processing of the plaintext proceeds **in three phases**.
- First, the 64-bit plaintext passes through an initial permutation (IP) that rearranges the bits to produce the permuted input.
- This is followed by a phase consisting of 16 rounds of the same function, which involves both permutation and substitution functions.
 - The output of the sixteenth round consists of 64 bits that are a function of the input plaintext and the key
 - The left and right halves of the output are swapped to produce the **preoutput**.

- Finally, the preoutput is passed through a permutation (IP-1) that is the inverse of the initial permutation function, to produce the 64-bit ciphertext.



Initial Permutation

- The initial permutation and its inverse are defined by tables
- The tables are to be interpreted as follows.
- The input to a table consists of 64 bits numbered from 1 to 64.
- The 64 entries in the permutation table contain a permutation of the numbers from 1 to 64.
- Each entry in the permutation table indicates the position of a numbered input bit in the output, which also consists of 64 bits

Permutation Tables for DES

Initial Permutation (IP)

58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

Expansion Permutation (E)

32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

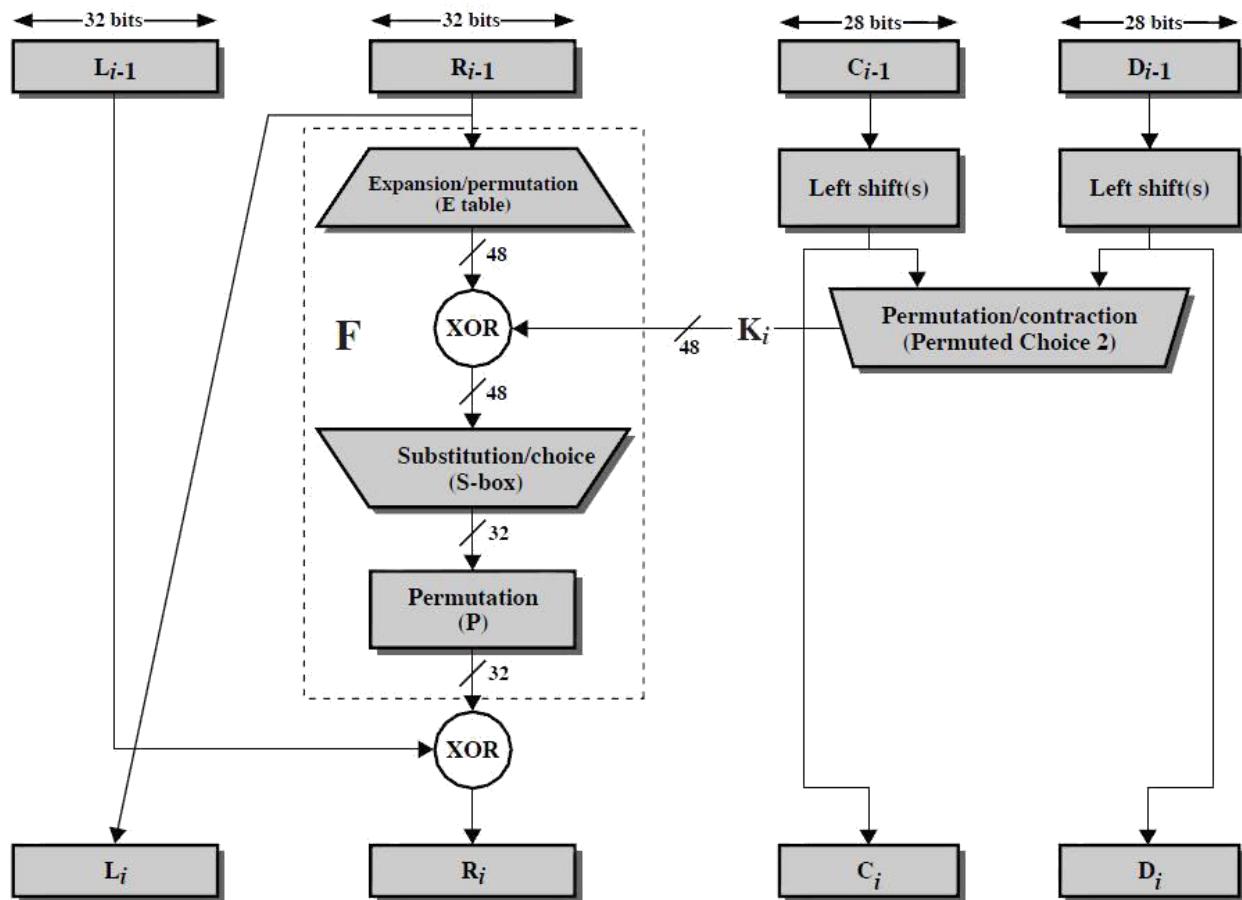
Inverse Initial Permutation (IP^{-1})

40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25

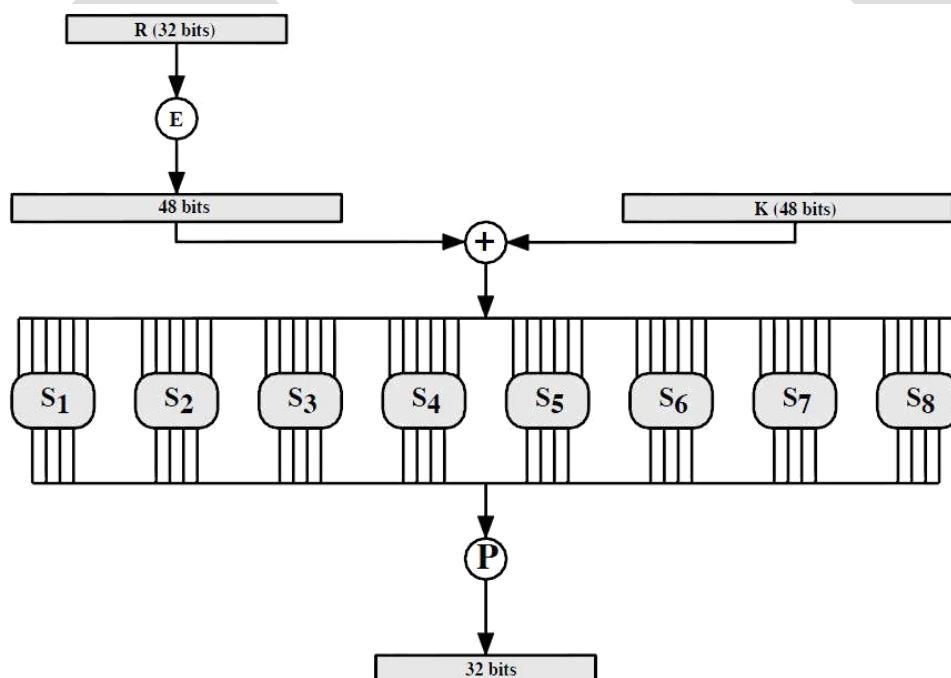
Permutation Function (P)

16	7	20	21	29	12	28	17
1	15	23	26	5	18	31	10
2	8	24	14	32	27	3	9
19	13	30	6	22	11	4	25

Details of Single Round



Calculation of $F(R, K)$



Definition of DES S-Boxes ($S_1 \dots S_8$)

14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

Key Generation

- a 64-bit key is used as input to the algorithm.
- The bits of the key are numbered from 1 through 64; every eighth bit is ignored
- The key is first subjected to a permutation governed by a table labeled Permuted Choice One
- The resulting 56-bit key is then treated as two 28-bit quantities, labeled C0 and D0.
- At each round, Ci-1 and Di-1 are separately subjected to a circular left shift, or rotation, of 1 or 2 bits
- These shifted values serve as input to the next round.
- They also serve as input to Permuted Choice Two , which produces a 48-bit output that serves as input to the function F(Ri-1, Ki).

DES Decryption

As with any Feistel cipher, decryption uses the same algorithm as encryption, except that the application of the subkeys is reversed

The Avalanche Effect

- a small change in either the plaintext or the key should produce a significant change in the ciphertext
- a change in one bit of the plaintext or one bit of the key should produce a change in many bits of the ciphertext
- DES exhibits a strong avalanche effect

Example

two plaintexts that differ by one bit were used

00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000

10000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000

with the key

0000001 1001011 0100100 1100010 0011100 0011000 0011100 0110010

- after just three rounds, 21 bits differ between the two blocks.
- On completion, the two ciphertexts differ in 34 bit positions

similar test in which a single plaintext is input with two keys that differ in only one bit position

01101000 10000101 00101111 01111010 00010011 01110110 11101011 10100100

Keys

1110010 1111011 1101111 0011000 0011101 0000100 0110001 11011100

0110010 1111011 1101111 0011000 0011101 0000100 0110001 11011100

about half of the bits in the ciphertext differ and that the avalanche effect is pronounced after just a few rounds

Differential and Linear Cryptanalysis

Differential Cryptanalysis

- powerful method to analyse block ciphers
- a statistical attack against Feistel ciphers
- uses cipher structure not previously used
- the analysis compares differences between two related encryptions, and looks for a known difference in leading to a known difference out with some (pretty small but still significant) probability.
- If a number of such differences are determined, it is feasible to determine the subkey used in the function f.
- compares two related pairs of encryptions with a
 - known difference in the input and
 - searching for a known difference in output when same subkeys are used

In differential cryptanalysis, we start with two messages, m and m' , with a known XOR difference $\Delta m = m \oplus m'$, and consider the difference between the intermediate message halves: $\Delta m_i = m_i \oplus m'_i$. Then we have

$$\begin{aligned}\Delta m_{i+1} &= m_{i+1} \oplus m'_{i+1} \\ &= [m_{i-1} \oplus f(m_i, K_i)] \oplus [m'_{i-1} \oplus f(m'_i, K_i)] \\ &= \Delta m_{i-1} \oplus [f(m_i, K_i) \oplus f(m'_i, K_i)]\end{aligned}$$

difference. Therefore, if we know Δm_{i-1} and Δm_i with high probability, then we know Δm_{i+1} with high probability. Furthermore, if a number of such differences are determined, it is feasible to determine the subkey used in the function f .

begin with two plaintext messages m and m'

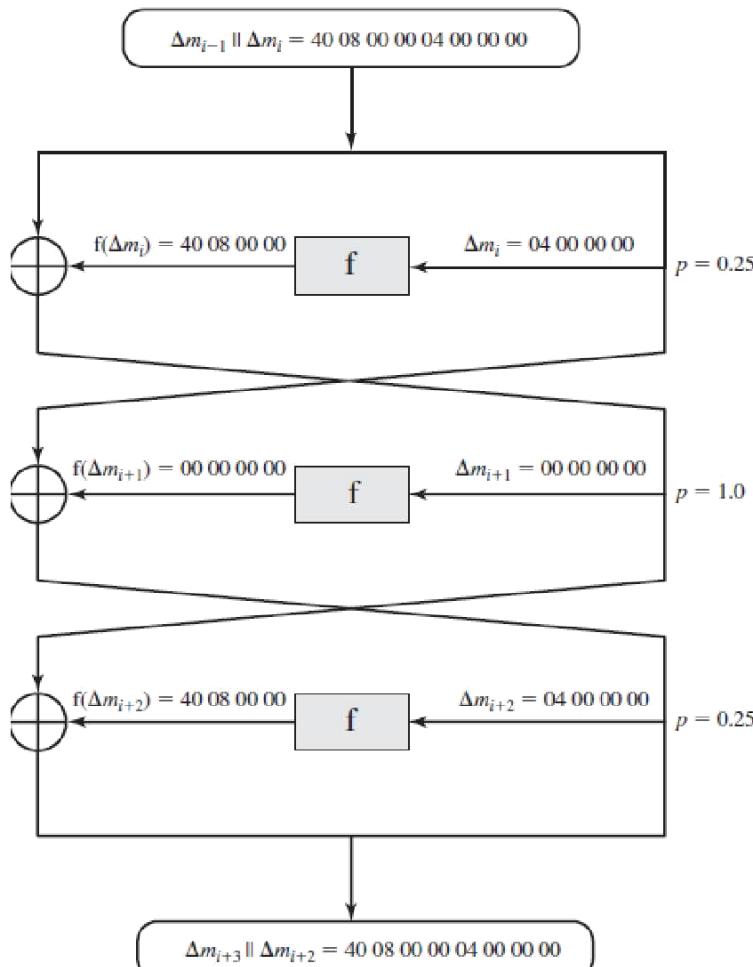
with a given difference and trace through a probable pattern of differences after each round to yield a probable difference for the ciphertext. Actually, there are two probable patterns of differences for the two 32-bit halves: $(\Delta m_{17} \parallel \Delta m_{16})$. Next, we submit m and m' for encryption to determine the actual difference under the unknown key and compare the result to the probable difference. If there is a match,

$$E(K, m) \oplus E(K, m') = (\Delta m_{17} \parallel \Delta m_{16})$$

then we suspect that all the probable patterns at all the intermediate rounds are correct. With that assumption, we can make some deductions about the key bits. This procedure must be repeated many times to determine all the key bits.

Differential Propagation through Three Rounds of DES

after three rounds, the probability that the output difference is as shown is equal to $0.25 * 1 * 0.25 = 0.0625$



Linear Cryptanalysis

- This attack is based on finding linear approximations to describe the transformations performed in DES
- This method can find a DES key given 2^{43} known plaintexts, as compared to 2^{47} chosen plaintexts for differential cryptanalysis
- it may be easier to acquire known plaintext rather than chosen plaintext
- infeasible as an attack on DES
- For a cipher with nbit plaintext and ciphertext blocks and an m-bit key, let the plaintext block be labeled $P[1], \dots, P[n]$, the cipher text block $C[1], \dots, C[n]$, and the key $K[1], \dots, K[m]$

Then define $A[i, j, \dots, k] = A[i] \oplus A[j] \oplus \dots \oplus A[k]$

The objective of linear cryptanalysis is to find an effective linear equation of the form

$$P[\alpha_1, \alpha_2, \dots, \alpha_a] \oplus C[\beta_1, \beta_2, \dots, \beta_b] = K[\gamma_1, \gamma_2, \dots, \gamma_c]$$

(where $x = 0$ or 1 ; $1 \leq a; b \leq n; c \leq m$; and where the α, β , and γ terms represent fixed, unique bit locations) that holds with probability $p \neq 0.5$. The further p is from

The further p is from 0.5 , the more effective the equation

Once a proposed relation is determined, the procedure is to compute the results of the lefthand side of the preceding equation for a large number of plaintext-ciphertext pairs

If the result is 0 more than half the time, assume $K[\gamma_1, \gamma_2, \dots, \gamma_c] = 0$.

If it is 1 most of the time, assume $K[\gamma_1, \gamma_2, \dots, \gamma_c] = 1$. This gives us a linear equation on the key bits.

Modes of operation

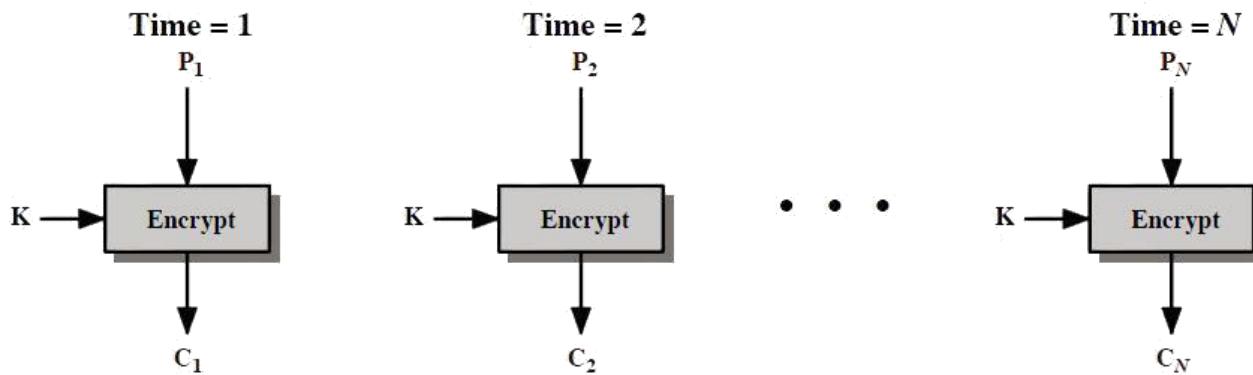
Mode	Description	Typical Application
Electronic Codebook (ECB)	Each block of 64 plaintext bits is encoded independently using the same key	Secure transmission of single values (e.g., an encryption key)
Cipher Block Chaining (CBC)	The input to the encryption algorithm is the XOR of the next 64 bits of plaintext and the preceding 64 bits of ciphertext	General-purpose block-oriented transmission Authentication
Cipher Feedback (CFB)	Input is processed j bits at a time. Preceding ciphertext is used as input to the encryption algorithm to produce pseudorandom output, which is XORed with plaintext to produce next unit of ciphertext	General-purpose stream-oriented transmission Authentication
Output Feedback (OFB)	Similar to CFB, except that the input to the encryption algorithm is the preceding DES output.	Stream-oriented transmission over noisy channel (e.g., satellite communication)
Counter (CTR)	Each block of plaintext is XORed with an encrypted counter. The counter is incremented for each subsequent block	General-purpose block-oriented transmission Useful for high-speed requirements

Various Modes

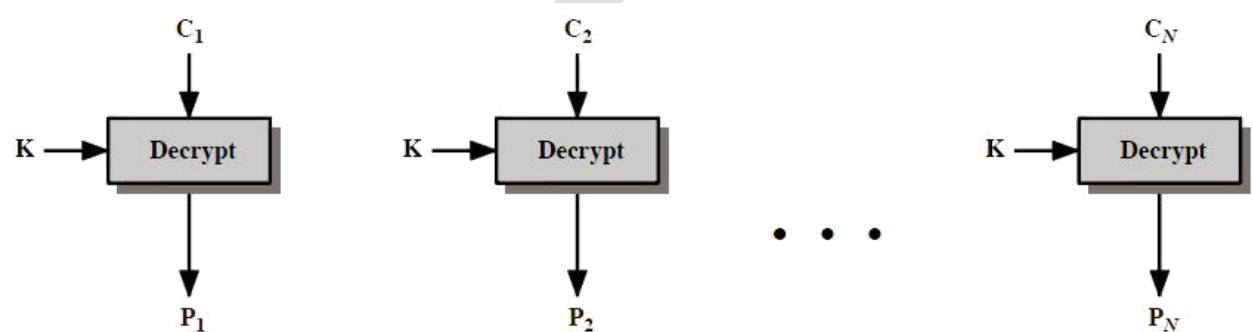
1. Electronic Codebook Mode
2. Cipher Block Chaining Mode
3. Cipher Feedback Mode
4. Output Feedback Mode
5. Counter Mode

Electronic Codebook Mode

Encryption

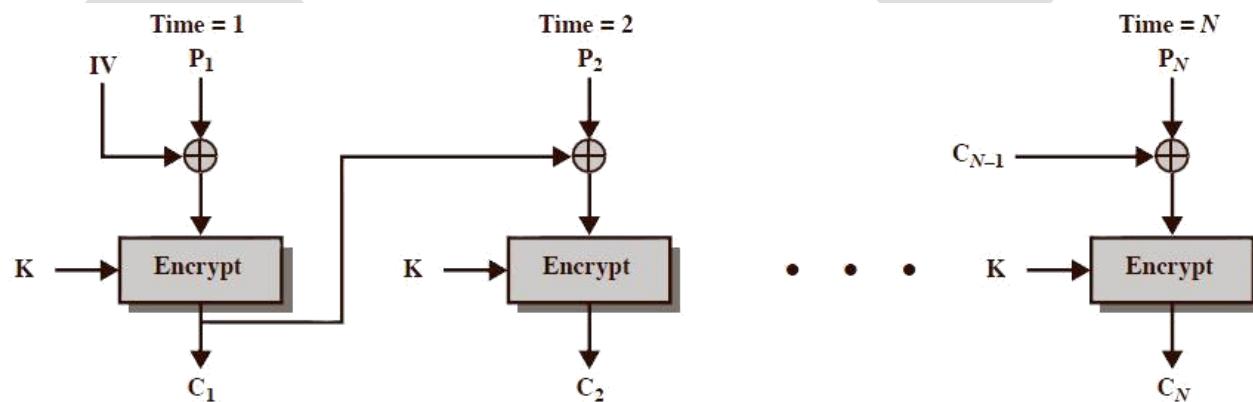


Decryption

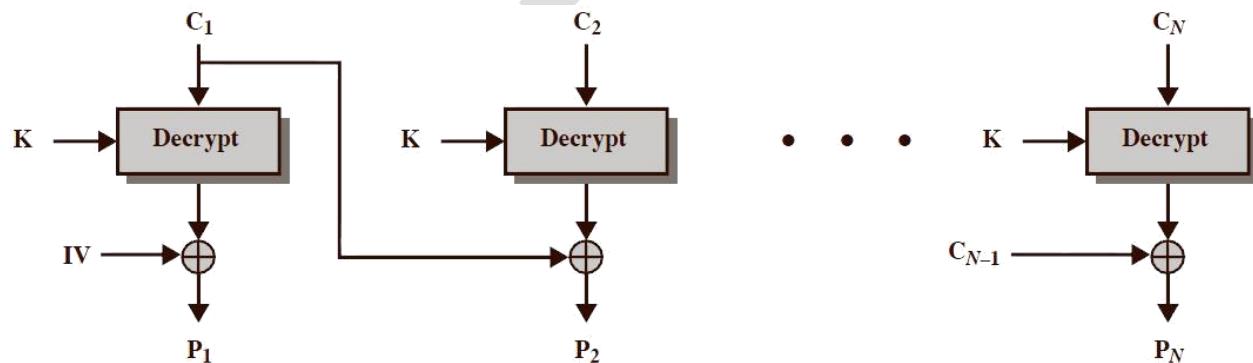


Cipher Block Chaining Mode

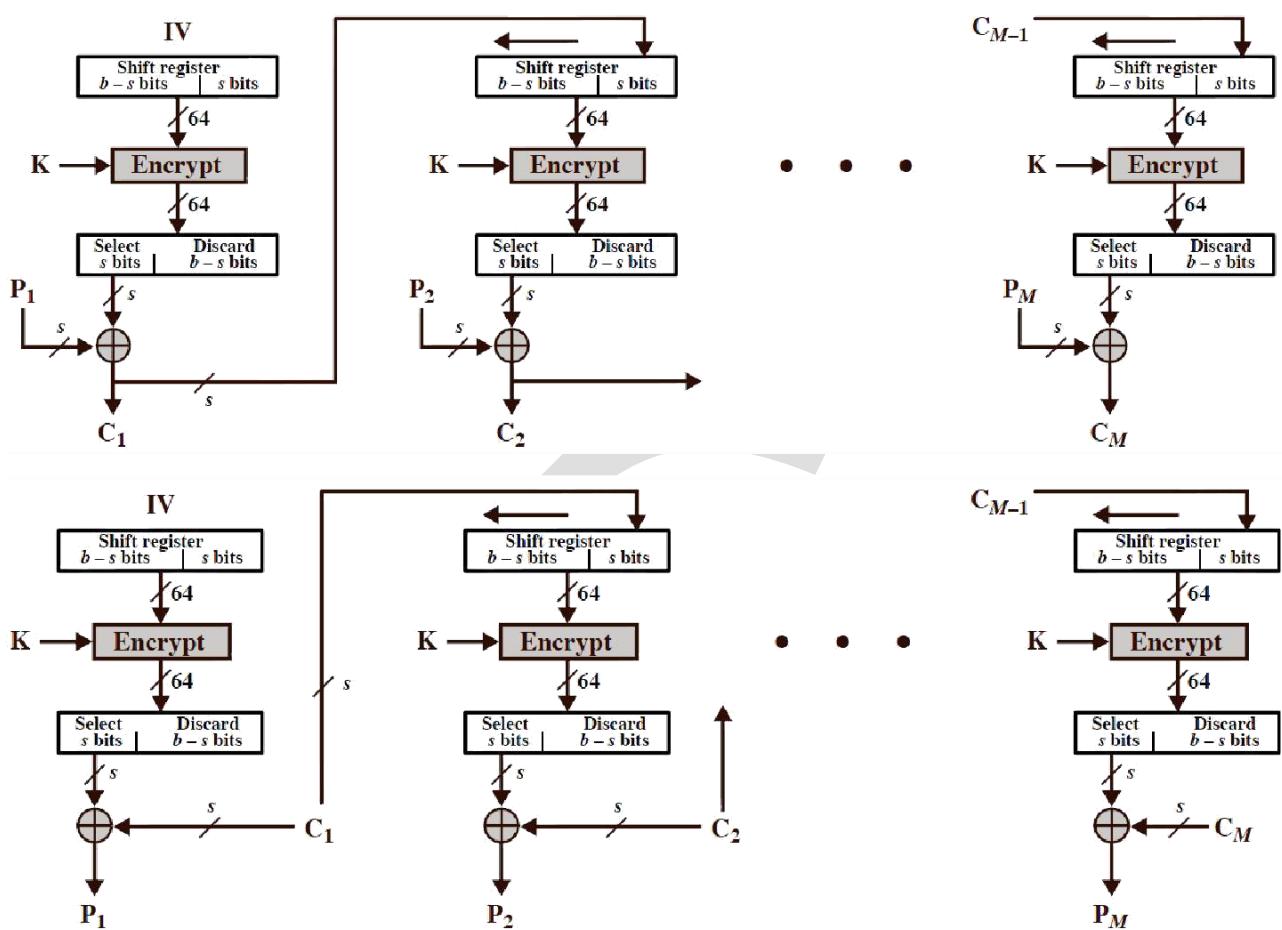
Encryption



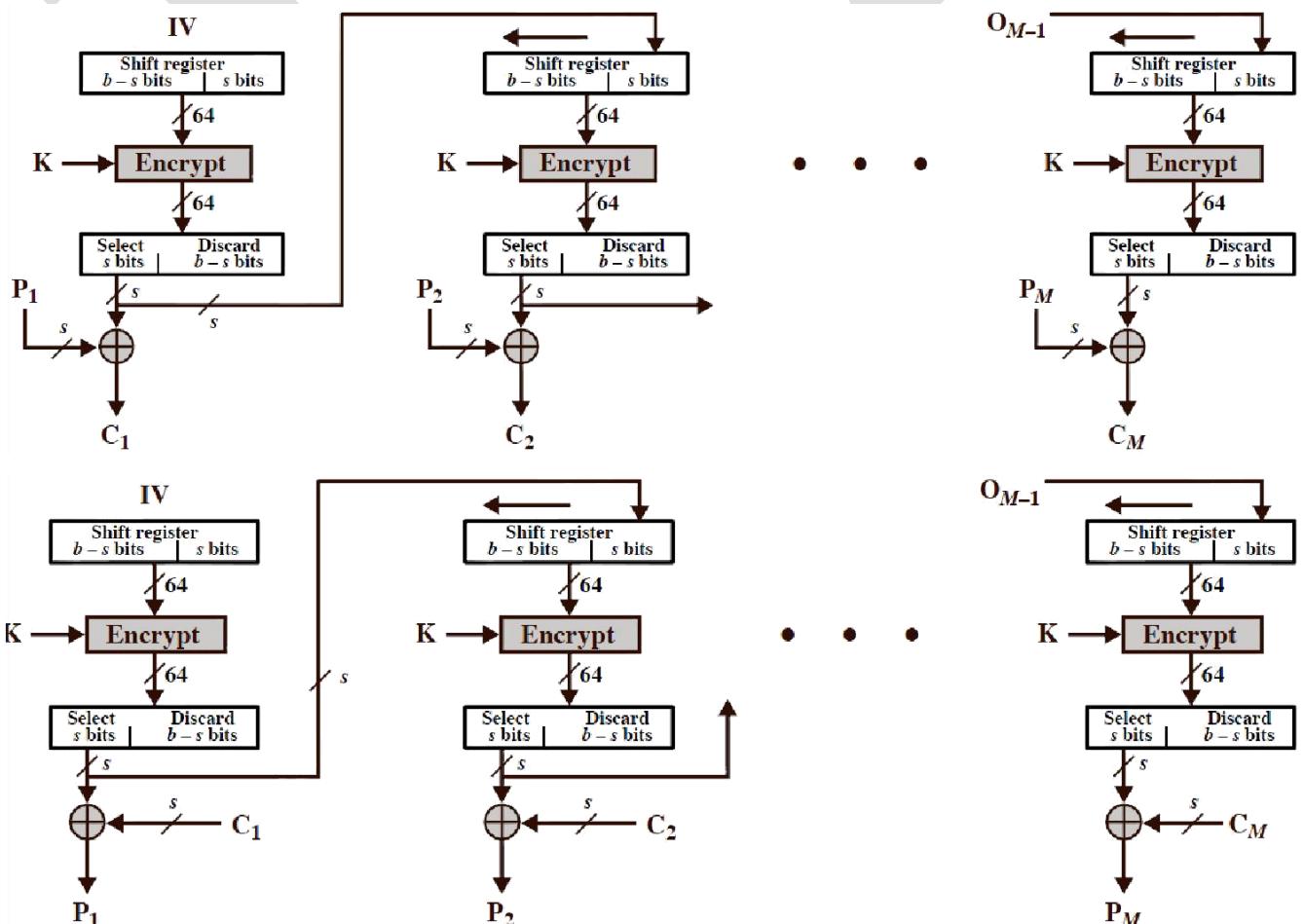
Decryption



Cipher Feedback Mode – Encryption / Decryption

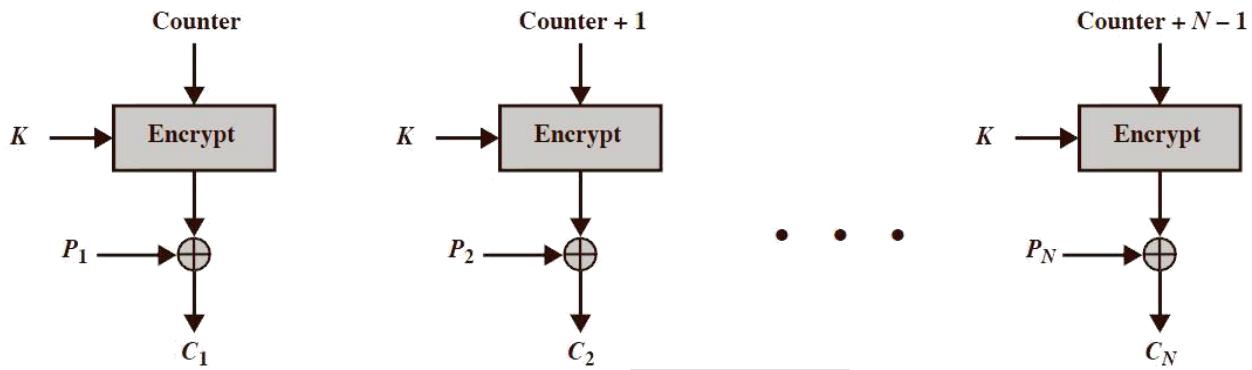


Output Feedback Mode– Encryption / Decryption

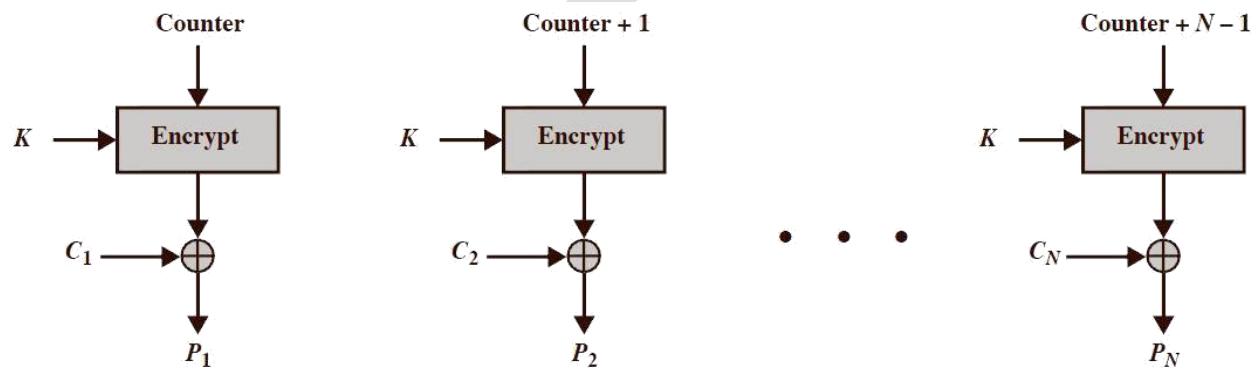


Counter Mode

Encryption



Decryption



- IV: initialization vector
- plaintext (padded as necessary) consists of a sequence of b-bit blocks, P_1, P_2, \dots, P_N ;
- the corresponding sequence of ciphertext blocks is C_1, C_2, \dots, C_N .
- the unit of transmission is s bits; a common value is s = 8

Advantages Of CTR Mode

- Hardware efficiency
- Software efficiency
- Preprocessing
- Random access
- Provable security
- Simplicity

Encryption Algorithms

Advanced Encryption Standard

- The AES Cipher
- AES Parameters
- AES Encryption and Decryption
- AES Data Structures
- AES Encryption Round
- Substitute Bytes Transformation
- ShiftRows Transformation
- AddRoundKey Transformation
- AES Key Expansion

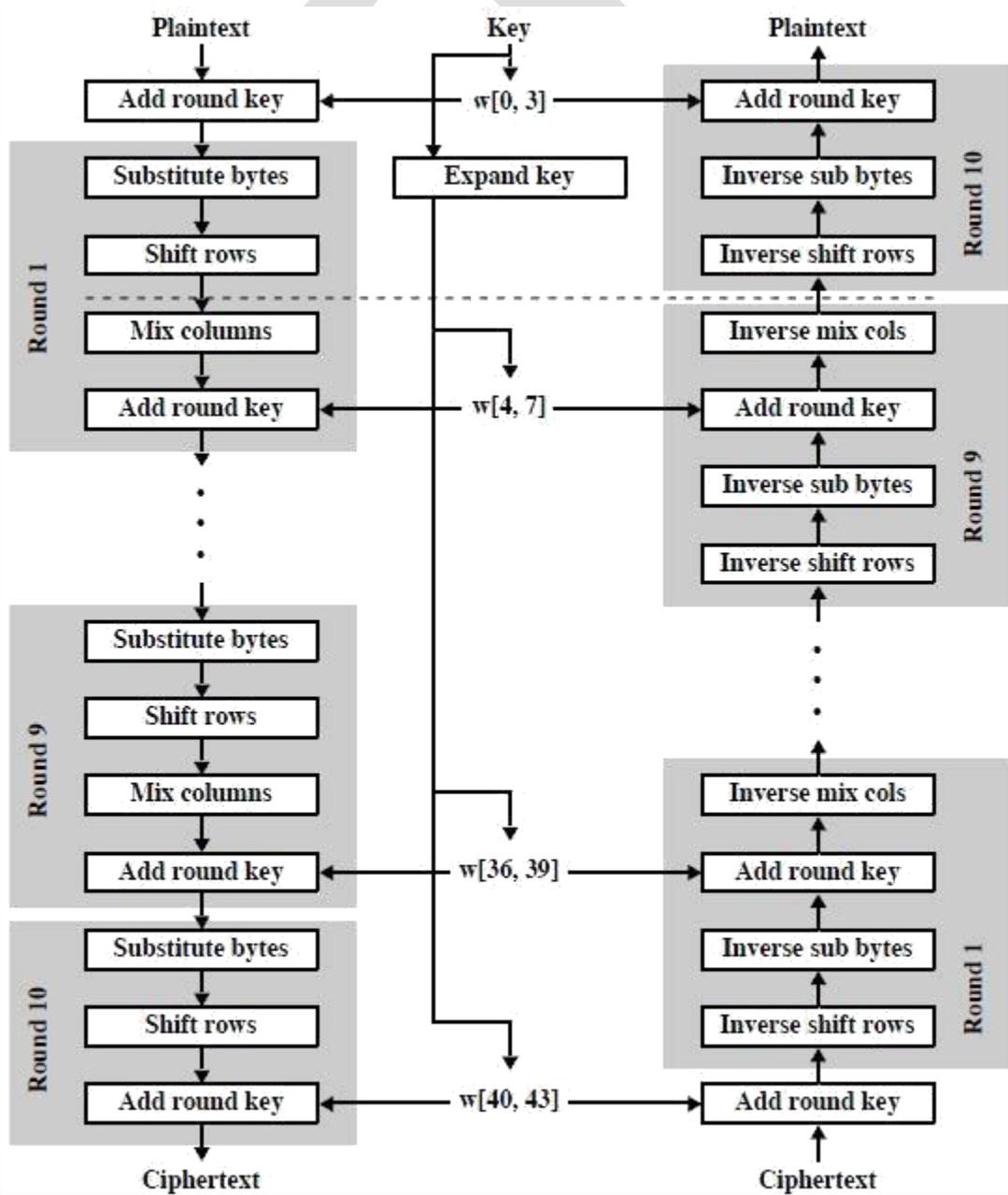
The AES Cipher

- The Rijndael proposal for AES defined a cipher in which the block length and the key length can be independently specified to be 128, 192, or 256 bits.
- The AES specification uses the same three key size alternatives but limits the block length to 128 bits.
- A number of AES parameters depend on the key length.
- In the description of this section, we assume a key length of 128 bits, which is likely to be the one most commonly implemented

AES Parameters

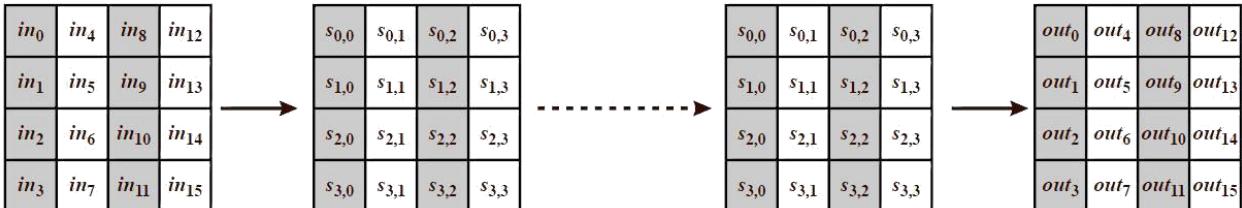
Key Size (words/bytes/bits)	4/16/128	6/24/192	8/32/256
Plaintext Block Size (words/bytes/bits)	4/16/128	4/16/128	4/16/128
Number of Rounds	10	12	14
Round Key Size (words/bytes/bits)	4/16/128	4/16/128	4/16/128
Expanded Key Size (words/bytes)	44/176	52/208	60/240

AES Encryption and Decryption



AES Data Structures

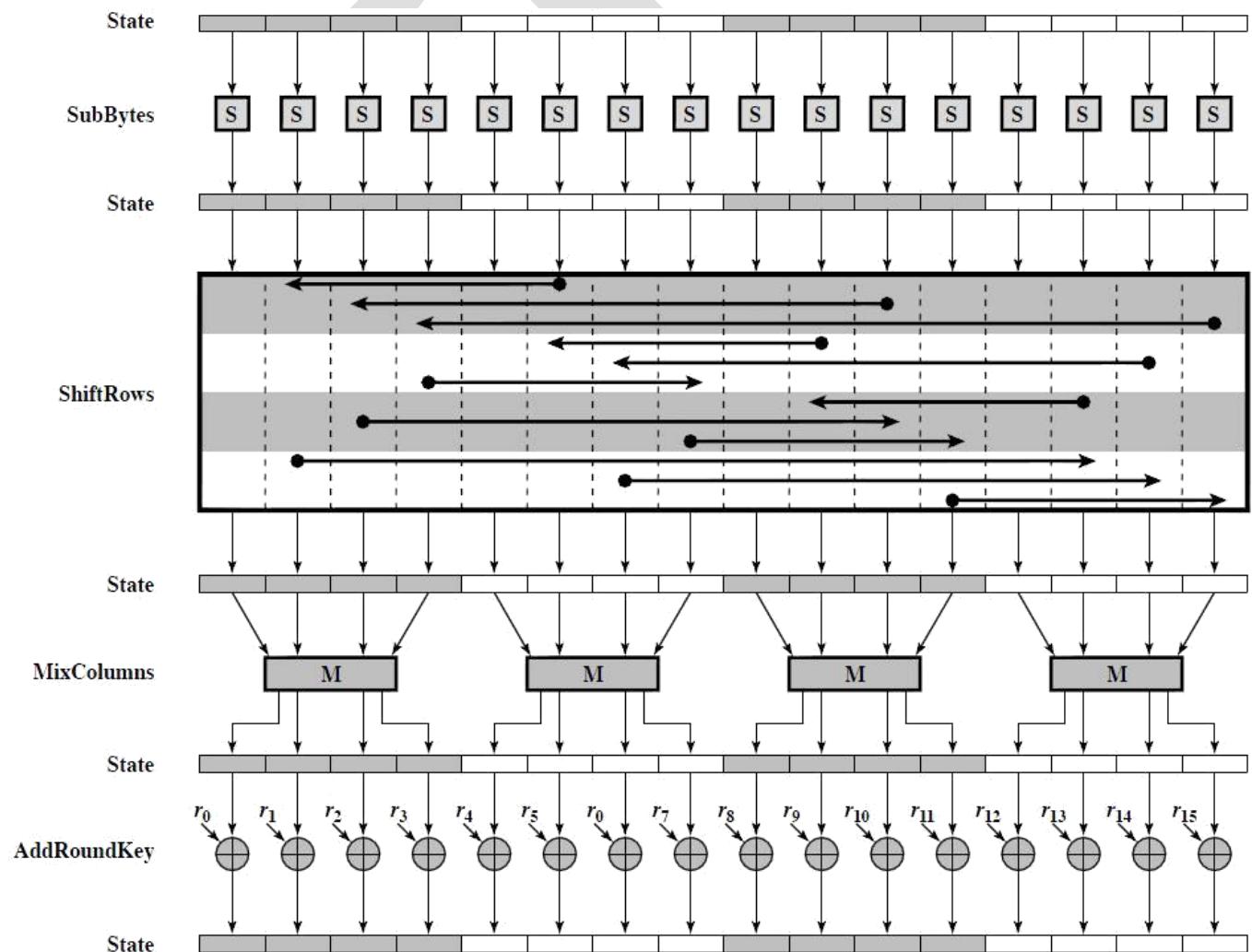
Input, state array, and output



Key and expanded key



AES Encryption Round



- Substitute Bytes Transformation
- ShiftRows Transformation
- AddRoundKey Transformation
- AES Key Expansion

Key Expansion Algorithm

The AES key expansion algorithm takes as input a four-word (16-byte) key and produces a linear array of 44 words (176 bytes). This is sufficient to provide a four-word round key for the initial AddRoundKey stage and each of the 10 rounds of the cipher. The pseudocode on the next page describes the expansion.

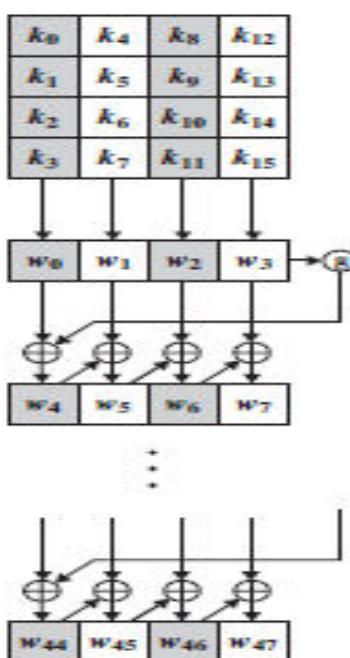
The key is copied into the first four words of the expanded key. The remainder of the expanded key is filled in four words at a time. Each added word $w[i]$ depends on the immediately preceding word, $w[i - 1]$, and the word four positions back, $w[i - 4]$. In three out of four cases, a simple XOR is used. For a word whose position in the w array is a multiple of 4, a more complex function is used. Figure 5.9 illustrates the generation of the expanded key, using the symbol g to represent that complex function. The function g consists of the following subfunctions.

1. RotWord performs a one-byte circular left shift on a word. This means that an input word $[B_0, B_1, B_2, B_3]$ is transformed into $[B_1, B_2, B_3, B_0]$.
2. SubWord performs a byte substitution on each byte of its input word, using the S-box (Table 5.2a).
3. The result of steps 1 and 2 is XORed with a round constant, $Rcon[j]$.

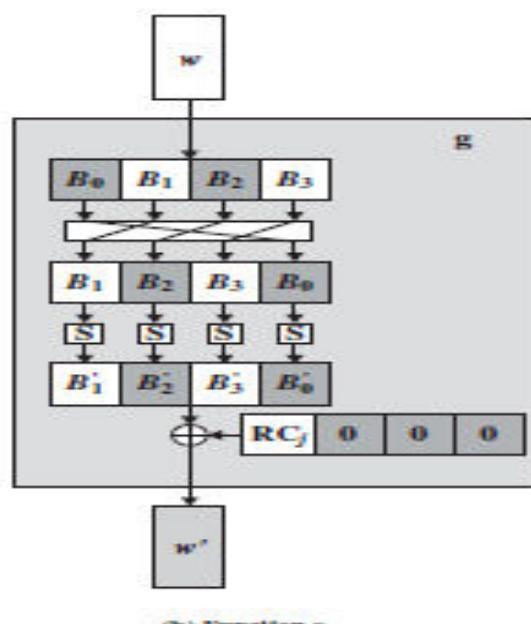
The round constant is a word in which the three rightmost bytes are always 0. Thus, the effect of an XOR of a word with $Rcon$ is to only perform an XOR on the left-most byte of the word. The round constant is different for each round and is defined as $Rcon[j] = (RC[j], 0, 0, 0)$, with $RC[1] = 1$, $RC[j] = 2 \cdot RC[j-1]$ and with multiplication defined over the field $GF(2^8)$. The values of $RC[j]$ in hexadecimal are

j	1	2	3	4	5	6	7	8	9	10
$RC[j]$	01	02	04	08	10	20	40	80	1B	36

```
KeyExpansion (byte key[16], word w[44])
{
    word temp
    for (i = 0; i < 4; i++)    w[i] = (key[4*i], key[4*i+1],
                                         key[4*i+2],
                                         key[4*i+3]);
    for (i = 4; i < 44; i++)
    {
        temp = w[i - 1];
        if (i mod 4 = 0)    temp = SubWord (RotWord (temp))
                           ⊕ Rcon[i/4];
        w[i] = w[i-4] ⊕ temp;
    }
}
```



(a) Overall algorithm

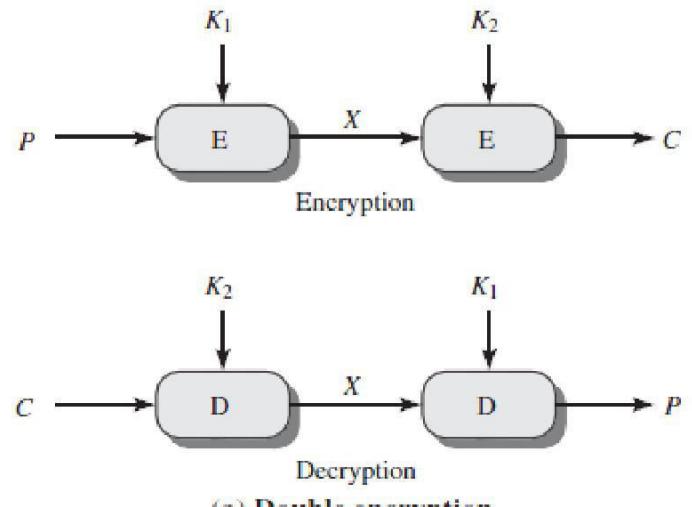


(b) Function g

Figure 5.9 AES Key Expansion

Double DES

- has two encryption stages and two keys
- Given a plaintext P and two encryption keys K₁ and K₂ and , ciphertext C is generated as $C = E(K_2, E(K_1, P))$
- Decryption requires that the keys be applied in reverse order $P = D(K_1, D(K_2, C))$
- this scheme apparently involves a key length of $56 * 2 = 112$ bits, resulting in a dramatic increase in cryptographic strength

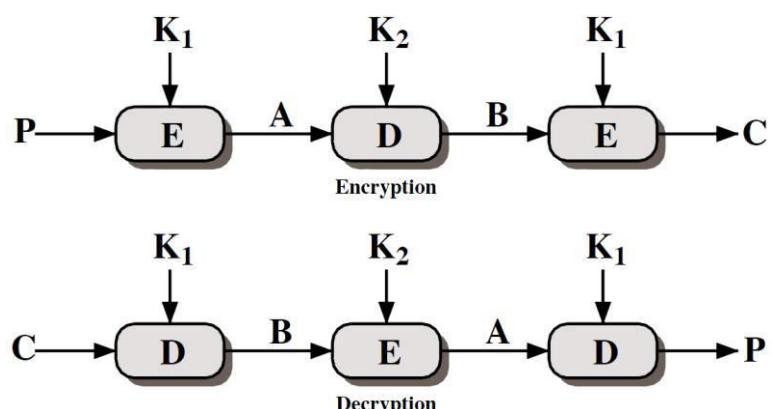


Meet-In-The-Middle Attack

- It is based on the observation that, if we have $C = E(K_2, E(K_1, P))$ then $X = E(K_1, P) = D(K_2, C)$
- Given a known pair, (P, C) the attack proceeds as follows
- First, encrypt P for all 2^{56} possible values of K₁
- Store these results in a table and then sort the table by the values of X
- Next, decrypt C using all 2^{56} possible values of K₂
- As each decryption is produced, check the result against the table for a match.
- If a match occurs, then test the two resulting keys against a new known plaintext–ciphertext pair.
- If the two keys produce the correct ciphertext, accept them as the correct keys.
- For any given plaintext P, there are 2^{64} possible ciphertext values that could be produced by double DES
- the foregoing procedure will produce about 2^{48} false alarms on the first (P,C) pair.
- With an additional 64 bits of known plaintext and ciphertext, the false alarm rate is reduced to $2^{48-64} = 2^{-16}$.
- If the meet-in-the-middle attack is performed on two blocks of known plaintext–ciphertext, the probability that the correct keys are determined is $1 - 2^{-16}$.
- The result is that a known plaintext attack will succeed against double DES, which has a key size of 112 bits, with an effort on the order of 2^{56} , which is not much more than the 2^{55} required for single DES

Triple DES

- triple encryption method that uses only two keys
- The function follows an encrypt-decrypt-encrypt (EDE) sequence
- $C = E(K_1, D(K_2, E(K_1, P)))$
- There is no cryptographic significance to the use of decryption for the second stage.
- advantage is that it allows users of 3DES to decrypt data encrypted by users of the older single DES:
- $C = E(K_1, D(K_1, E(K_1, P))) = E(K_1, P)$



Attacks on TDES

Known-Plaintext Attack on Triple DES

Triple DES with Three Keys

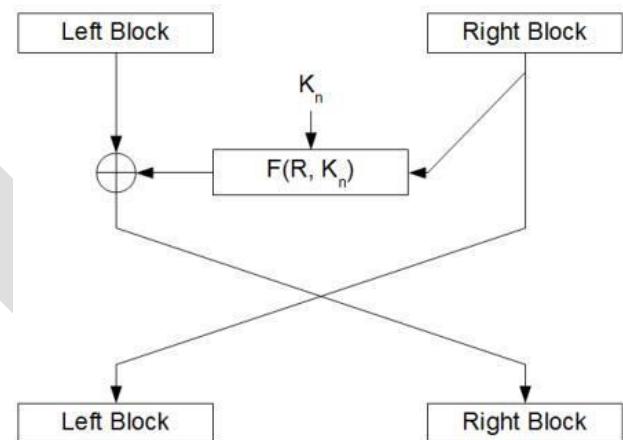
- Three-key 3DES has an effective key length of 168 bits and is defined as follows:
- $C = E(K_3, D(K_2, E(K_1, P)))$
- Backward compatibility with DES is provided by putting K₃ = K₂ or K₁ = K₂.
- A number of Internet-based applications have adopted three-key 3DES, including PGP and S/MIME

Blowfish

- Symmetric block cipher that can be effectively used for encryption and safeguarding of data
- It takes a variable-length key, from 32 bits to 448 bits, making it ideal for securing data.
- fast, free alternative to existing encryption algorithms
- unpatented and license-free, and is available free for all uses
- **Blowfish Algorithm** is a Feistel Network, iterating a simple encryption function 16 times.
- The block size is 64 bits, and the key can be any length up to 448 bits.
- Although there is a complex initialization phase required before any encryption can take place, the actual encryption of data is very efficient on large microprocessors.
- Blowfish is a variable-length key block cipher.
- It is suitable for applications where the key does not change often, like a communications link or an automatic file encryptor.
- It is significantly faster than most encryption algorithms when implemented on 32-bit microprocessors with large data caches

Feistel Networks

- A Feistel network is a general method of transforming any function (usually called an Ffunction) into a permutation.
- It was invented by Horst Feistel and has been used in many block cipher designs.
- The working of a Feistel Network is given below:
 - Split each block into halves
 - Right half becomes new left half
 - New right half is the final result when the left half is XOR'd with the result of applying f to the right half and the key.
 - Note that previous rounds can be derived even if the function f is not invertible



The Blowfish Algorithm:

- Manipulates data in large blocks
- Has a 64-bit block size.
- Has a scalable key, from 32 bits to at least 256 bits.
- Uses simple operations that are efficient on microprocessors.
 - e.g., exclusive-or, addition, table lookup, modular- multiplication.
 - It does not use variable-length shifts or bit-wise permutations, or conditional jumps.
- Employs precomputable subkeys.
 - On large-memory systems, these subkeys can be precomputed for faster operation.
 - Not precomputing the subkeys will result in slower operation, but it should still be possible to encrypt data without any precomputations.
- Consists of a variable number of iterations.
- Uses subkeys that are a one-way hash of the key.
 - This allows the use of long passphrases for the key without compromising security.
- Has no linear structures that reduce the complexity of exhaustive search.
- Uses a design that is simple to understand.

Description Of The Algorithm

- Blowfish is a variable-length key, 64-bit block cipher.
- The algorithm consists of two parts:
 - a key-expansion part and
 - a data- encryption part.
- Key expansion converts a key of at most 448 bits into several subkey arrays totaling 4168 bytes.
- Data encryption occurs via a 16-round Feistel network.
- Each round consists of a keydependent permutation, and a key- and data-dependent substitution.
- All operations are XORs and additions on 32-bit words.

- The only additional operations are four indexed array data lookups per round

Subkeys

- Blowfish uses a large number of subkeys.
- These keys must be precomputed before any data encryption or decryption.
- The P-array consists of 18 32-bit subkeys:P1, P2,..., P18.
- There are four 32-bit S-boxes with 256 entries each:
 - S1,0, S1,1,..., S1,255;
 - S2,0, S2,1,..., S2,255;
 - S3,0, S3,1,..., S3,255;
 - S4,0, S4,1,..., S4,255.

Encryption

- Blowfish has 16 rounds.
- The input is a 64-bit data element, x.
- Divide x into two 32-bit halves: xL, xR.
- Then,

```
for i = 1 to 16:  
    xL = xL XOR Pi  
    xR = F(xL) XOR xR  
    Swap xL and xR
```

- After the sixteenth round, swap xL and xR again to undo the last swap.
- Then, xR = xR XOR P17 and xL = xL XOR P18.
- Finally, recombine xL and xR to get the ciphertext.

Decryption

- Exactly the same as encryption, except that P1, P2,..., P18 are used in the reverse order

Generating the Subkeys

The subkeys are calculated using the Blowfish algorithm:

1. Initialize first the P-array and then the four S-boxes, in order, with a fixed string.

This string consists of the hexadecimal digits of pi (less the initial 3): P1 = 0x243f6a88, P2 = 0x85a308d3, P3 = 0x13198a2e, P4 = 0x03707344, etc.

2. XOR P1 with the first 32 bits of the key, XOR P2 with the second 32-bits of the key, and so on for all bits of the key (possibly up to P14). Repeatedly cycle through the key bits until the entire P-array has been XORED with key bits. (For every short key, there is at least one equivalent longer key; for example, if A is a 64-bit key, then AA, AAA, etc., are equivalent keys.)

3. Encrypt the all-zero string with the Blowfish algorithm, using the subkeys described in steps (1) and (2).

4. Replace P1 and P2 with the output of step (3).

5. Encrypt the output of step (3) using the Blowfish algorithm with the modified subkeys.

6. Replace P3 and P4 with the output of step (5).

7. Continue the process, replacing all entries of the P array, and then all four S-boxes in order, with the output of the continuously changing Blowfish algorithm.

- In total, 521 iterations are required to generate all required subkeys.
- Applications can store the subkeys rather than execute this derivation process multiple times.

RC5

Introduction

- a proprietary cipher owned by RSADSI
- designed by Ronald Rivest (of RSA fame)
- used in various RSADSI products
- can vary key size / data size / no rounds
- very clean and simple design
- easy implementation on various CPUs
- yet still regarded as secure

RC5 Ciphers

- RC5 is a family of ciphers RC5-w/r/b
 - w = word size in bits (16/32/64) nb data=2w
 - r = number of rounds (0..255)
 - b = number of bytes in key (0..255)
- nominal version is RC5-32/12/16
 - 32-bit words so encrypts 64-bit data blocks
 - using 12 rounds
 - with 16 bytes (128-bit) secret key

RC5 Key Expansion

- RC5 uses $2r+2$ subkey words (w -bits)
- subkeys are stored in array $S[i]$, $i=0..t-1$
- the key schedule consists of
 - initializing S to a fixed pseudorandom value, based on constants e and ϕ
 - the byte key is copied (little-endian) into a c -word array L
 - a mixing operation then combines L and S to form the final S array

RC5 Encryption

- split input into two halves A & B
- $L_0 = A + S[0];$
- $R_0 = B + S[1];$
- for $i = 1$ to r do
 - $L_i = ((L_{i-1} \text{ XOR } R_{i-1}) \lll R_{i-1}) + S[2 \times i];$
 - $R_i = ((R_{i-1} \text{ XOR } L_i) \lll L_i) + S[2 \times i + 1];$
- each round is like 2 DES rounds
- note rotation is main source of non-linearity
- need reasonable number of rounds (eg 12-16)

RC5 Modes

- RFC2040 defines 4 modes used by RC5
- RC5 Block Cipher, is ECB mode
- RC5-CBC, is CBC mode
- RC5-CBC-PAD, is CBC with padding by bytes with value being the number of padding bytes
- RC5-CTS, a variant of CBC which is the same size as the original message, uses ciphertext stealing to keep size same as original.

Public Key Cryptography

Principles of Public-Key Cryptosystems

- Public-Key Cryptosystems
- Applications for Public-Key Cryptosystems
- Requirements for Public-Key Cryptography
- Public-Key Cryptanalysis

Public-Key Cryptosystems

Introduction

- The concept evolved from an attempt to attack two of the most difficult problems associated with symmetric encryption
 - Key Distribution
 - The Digital Signatures
- Called as Asymmetric Cryptography
- Asymmetric algorithms make use of one key for encryption, another for decryption

Characteristics of Asymmetric algorithms

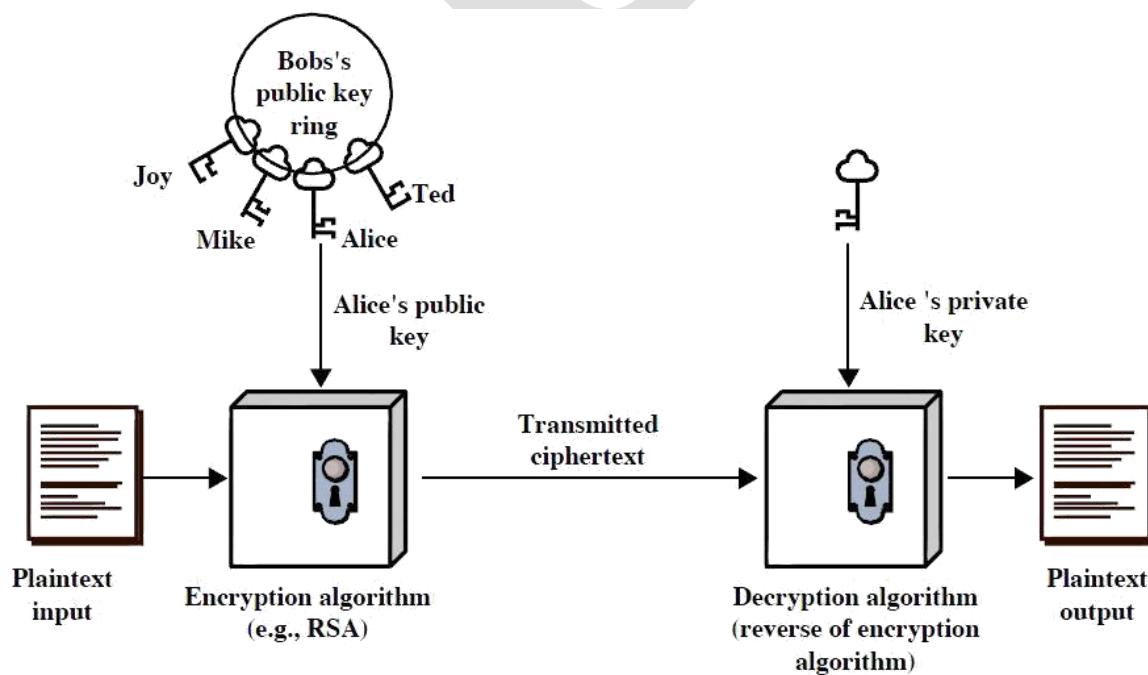
- It is computationally infeasible to determine the decryption key given only knowledge of the cryptographic algorithm and the encryption key
- Either of the two related keys can be used for encryption, with the other used for decryption

Public-Key Cryptography

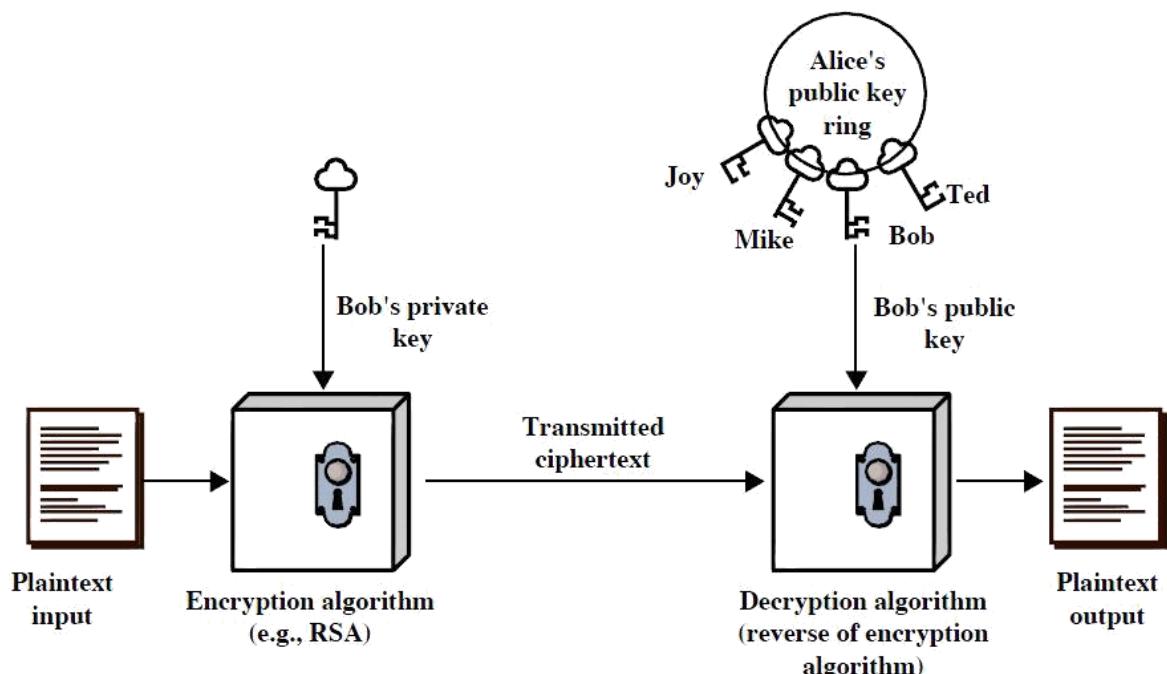
Six Ingredients

- **Plaintext:** This is the readable message or data that is fed into the algorithm as input.
- **Encryption algorithm:** The encryption algorithm performs various transformations on the plaintext.
- **Public and private keys:** This is a pair of keys that have been selected so that if one is used for encryption, the other is used for decryption. The exact transformations performed by the algorithm depend on the public or private key that is provided as input.
- **Ciphertext:** This is the scrambled message produced as output. It depends on the plaintext and the key. For a given message, two different keys will produce two different ciphertexts.
- **Decryption algorithm:** This algorithm accepts the ciphertext and the matching key and produces the original plaintext.

Encryption



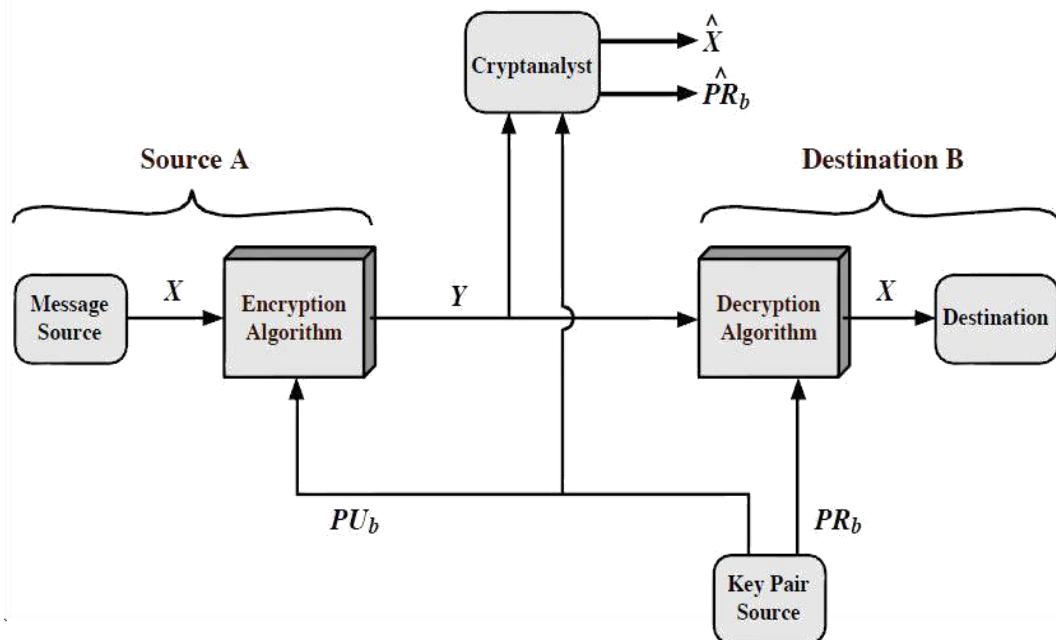
Authentication



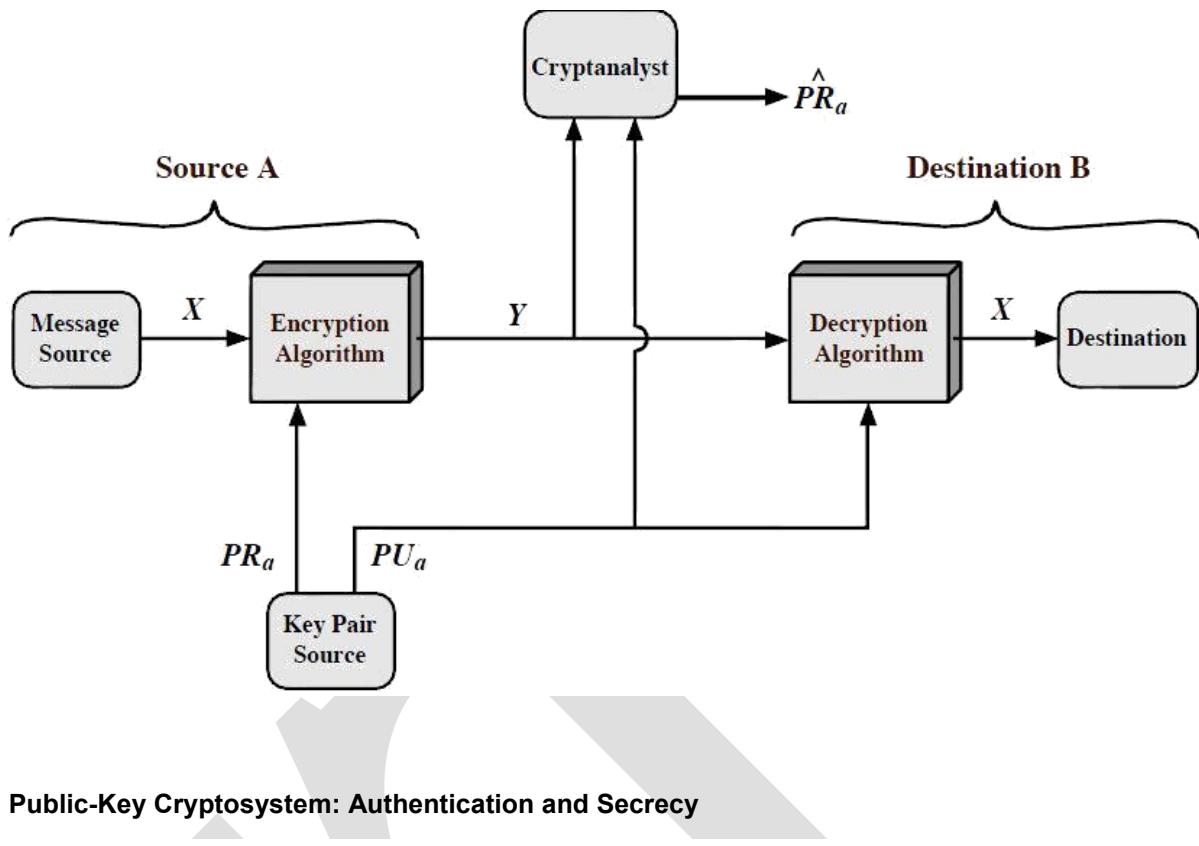
Comparison with Symmetric Key Encryption

Conventional Encryption	Public-Key Encryption
<p><i>Needed to Work</i></p> <ol style="list-style-type: none"> 1. The same algorithm with the same key is used for encryption and decryption. 2. The sender and receiver must share the algorithm and the key. 	<p><i>Needed to Work</i></p> <ol style="list-style-type: none"> 1. One algorithm is used for encryption and decryption with a pair of keys, one for encryption and one for decryption. 2. The sender and receiver must each have one of the matched pair of keys (not the same one).
<p><i>Needed for Security:</i></p> <ol style="list-style-type: none"> 1. The key must be kept secret. 2. It must be impossible or at least impractical to decipher a message if no other information is available. 3. Knowledge of the algorithm plus samples of ciphertext must be insufficient to determine the key. 	<p><i>Needed for Security:</i></p> <ol style="list-style-type: none"> 1. One of the two keys must be kept secret. 2. It must be impossible or at least impractical to decipher a message if no other information is available. 3. Knowledge of the algorithm plus one of the keys plus samples of ciphertext must be insufficient to determine the other key.

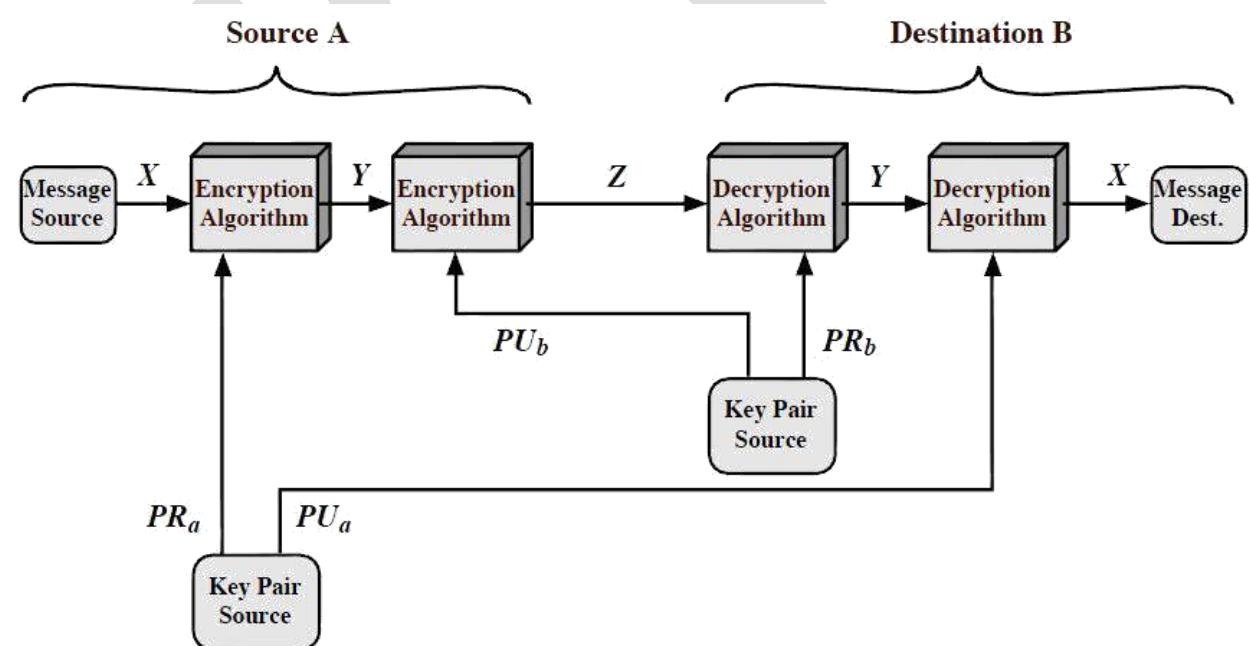
Public-Key Cryptosystem: Secrecy



Public-Key Cryptosystem: Authentication



Public-Key Cryptosystem: Authentication and Secrecy



Applications for Public-Key Cryptosystems

Encryption/decryption

The sender encrypts a message with the recipient's public key.

Digital signature

The sender "signs" a message with its private key. Signing is achieved by a cryptographic algorithm applied to the message or to a small block of data that is a function of the message.

Key exchange

Two sides cooperate to exchange a session key. Several different approaches are possible, involving the private key(s) of one or both parties.

- original message, M.
- The two keys can be applied in either order: $M = D[PUb, E(PRb, M)] = D[PRb, E(PUb, M)]$

Public-Key Cryptanalysis

Three types of attacks

- Brute force
- Deducing the private key
- Probale message attack

Brute force

- public-key encryption scheme is vulnerable to a brute-force attack
- countermeasure is to use large keys
- Public-key systems depend on the use of some sort of invertible mathematical function
- the key size must be large enough to make brute-force attack impractical but small enough for practical encryption and decryption

Deducing the private key

- find some way to compute the private key given the public key
- So far, not been mathematically proven that this is infeasible for a particular public-key algorithm
- Not been successful till date

Probale message attack

- peculiar to public-key systems
- Suppose, for example, that a message were to be sent that consisted solely of a 56-bit DES key.
- An adversary could encrypt all possible 56-bit DES keys using the public key and could discover the encrypted key by matching the transmitted ciphertext.
- Thus, no matter how large the key size of the public-key scheme, the attack is reduced to a brute-force attack on a 56-bit key.
- This attack can be thwarted by appending some random bits to such simple messages

Rivest-Shamir-Adleman (RSA) Algorithm

- 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 decimal digits
- public-key encryption algorithm with a public key of $PU = \{e, n\}$ and a private key of $PR = \{d, n\}$.

Key Generation

Select p, q	p and q both prime, $p \neq q$
Calculate $n = p \times q$	
Calcuate $\phi(n) = (p - 1)(q - 1)$	
Select integer e	$\gcd(\phi(n), e) = 1; 1 < e < \phi(n)$
Calculate d	$d \equiv e^{-1} \pmod{\phi(n)}$
Public key	$PU = \{e, n\}$
Private key	$PR = \{d, n\}$

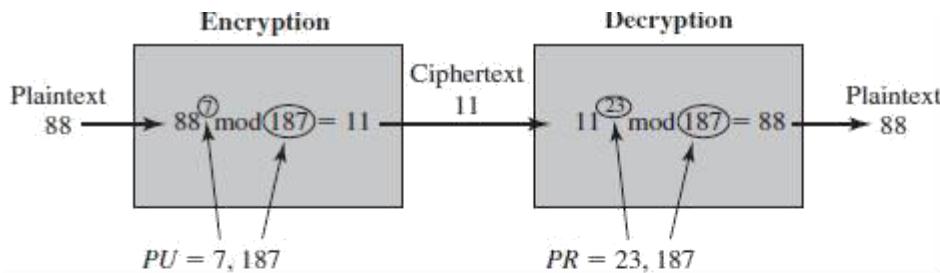
Encryption

Plaintext:	$M < n$
Ciphertext:	$C = M^e \pmod{n}$

Decryption

Ciphertext:	C
Plaintext:	$M = C^d \pmod{n}$

Example of RSA Algorithm



1. Select two prime numbers, $p = 17$ and $q = 11$.
2. Calculate $n = pq = 17 \times 11 = 187$.
3. Calculate $\phi(n) = (p - 1)(q - 1) = 16 \times 10 = 160$.
4. Select e such that e is relatively prime to $\phi(n) = 160$ and less than $\phi(n)$; we choose $e = 7$.
5. Determine d such that $de \equiv 1 \pmod{160}$ and $d < 160$. The correct value is $d = 23$, because $23 \times 7 = 161 = (1 \times 160) + 1$; d can be calculated using the extended Euclid's algorithm (Chapter 4).

The resulting keys are public key $PU = \{7, 187\}$ and private key $PR = \{23, 187\}$.

Encryption

we need to calculate $C = 88^7 \text{ mod } 187$.

$$88^7 \text{ mod } 187 = [(88^4 \text{ mod } 187) \times (88^2 \text{ mod } 187) \times (88^1 \text{ mod } 187)] \text{ mod } 187$$

$$88^1 \text{ mod } 187 = 88$$

$$88^2 \text{ mod } 187 = 7744 \text{ mod } 187 = 77$$

$$88^4 \text{ mod } 187 = 59,969,536 \text{ mod } 187 = 132$$

$$88^7 \text{ mod } 187 = (88 \times 77 \times 132) \text{ mod } 187 = 894,432 \text{ mod } 187 = 11$$

Decryption

For decryption, we calculate $M = 11^{23} \text{ mod } 187$:

$$11^{23} \text{ mod } 187 = [(11^1 \text{ mod } 187) \times (11^2 \text{ mod } 187) \times (11^4 \text{ mod } 187) \times (11^8 \text{ mod } 187) \times (11^8 \text{ mod } 187)] \text{ mod } 187$$

$$11^1 \text{ mod } 187 = 11$$

$$11^2 \text{ mod } 187 = 121$$

$$11^4 \text{ mod } 187 = 14,641 \text{ mod } 187 = 55$$

$$11^8 \text{ mod } 187 = 214,358,881 \text{ mod } 187 = 33$$

$$11^{23} \text{ mod } 187 = (11 \times 121 \times 55 \times 33 \times 33) \text{ mod } 187 = 79,720,245 \text{ mod } 187 = 88$$

The Security of RSA

Four possible approaches to attacking the RSA algorithm are as follows:

Brute force

- This involves trying all possible private keys.

- The defense is to use a large key space
- the larger the number of bits in e and d, the better
- the larger the size of the key, the slower the system will run

Mathematical attacks

There are several approaches, all equivalent in effort to factoring the product of two primes.

Timing attacks

These depend on the running time of the decryption algorithm.

Chosen ciphertext attacks

This type of attack exploits properties of the RSA algorithm.

Mathematical Attacks

- Three approaches to attacking RSA mathematically:

1. Factor n into its two prime factors. This enables calculation of $\phi(n) = (p - 1) \times (q - 1)$, which in turn enables determination of $d \equiv e^{-1} \pmod{\phi(n)}$.
2. Determine $\phi(n)$ directly, without first determining p and q . Again, this enables determination of $d \equiv e^{-1} \pmod{\phi(n)}$.
3. Determine d directly, without first determining $\phi(n)$.

To avoid values of n that may be factored more easily, the algorithm's inventors suggest the following constraints on p and q .

1. p and q should differ in length by only a few digits. Thus, for a 1024-bit key (309 decimal digits), both p and q should be on the order of magnitude of 10^{75} to 10^{100} .
2. Both $(p - 1)$ and $(q - 1)$ should contain a large prime factor.
3. $\gcd(p - 1, q - 1)$ should be small.

if $e < n$ and $d < n^{1/4}$, then d can be easily determined

Timing Attack

- This attack is alarming for two reasons:
 - It comes from a completely unexpected direction
 - it is a ciphertext-only attack
- A timing attack is somewhat analogous to a burglar guessing the combination of a safe by observing how long it takes for someone to turn the dial from number to number.
- We can explain the attack using the modular exponentiation algorithm
- modular exponentiation is accomplished bit by bit, with one modular multiplication performed at each iteration and an additional modular multiplication performed for each 1 bit

Working of this attack

- The attack proceeds bit-by-bit starting with the leftmost bit, b_k
- Suppose that the first j bits are known
- For a given ciphertext, the attacker can complete the first j iterations of the for loop.
- The operation of the subsequent step depends on the unknown exponent bit.
- if the observed time to execute the decryption algorithm is always slow when this particular iteration is slow with a 1 bit, then this bit is assumed to be 1.
- If a number of observed execution times for the entire algorithm are fast, then this bit is assumed to be 0

Methods to overcome timing attacks

Constant exponentiation time

- Ensure that all exponentiations take the same amount of time before returning a result.
- This is a simple fix but does degrade performance

Random delay

Better performance could be achieved by adding a random delay to the exponentiation algorithm to confuse the timing attack.

Blinding

- Multiply the ciphertext by a random number before performing exponentiation.
- This process prevents the attacker from knowing what ciphertext bits are being processed inside the computer and therefore prevents the bit-by-bit analysis essential to the timing attack

Cryptography and Network Security



Dr. B. Shanthini
Prof. & Head, IT Department
St. Peter's College of Engineering & Technology

Unit 3 – Outline

- Authentication requirement
- Authentication functions
- MAC
- Hash function
- Security of hash function and MAC
- MD5 and SHA
- HMAC and CMAC
- Digital signature and authentication protocols
- DSS – El Gamal – Schnorr

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
- **Three functions used:**
 - Message Encryption
 - Message Authentication Code (MAC)
 - Hash Function

Security Requirements

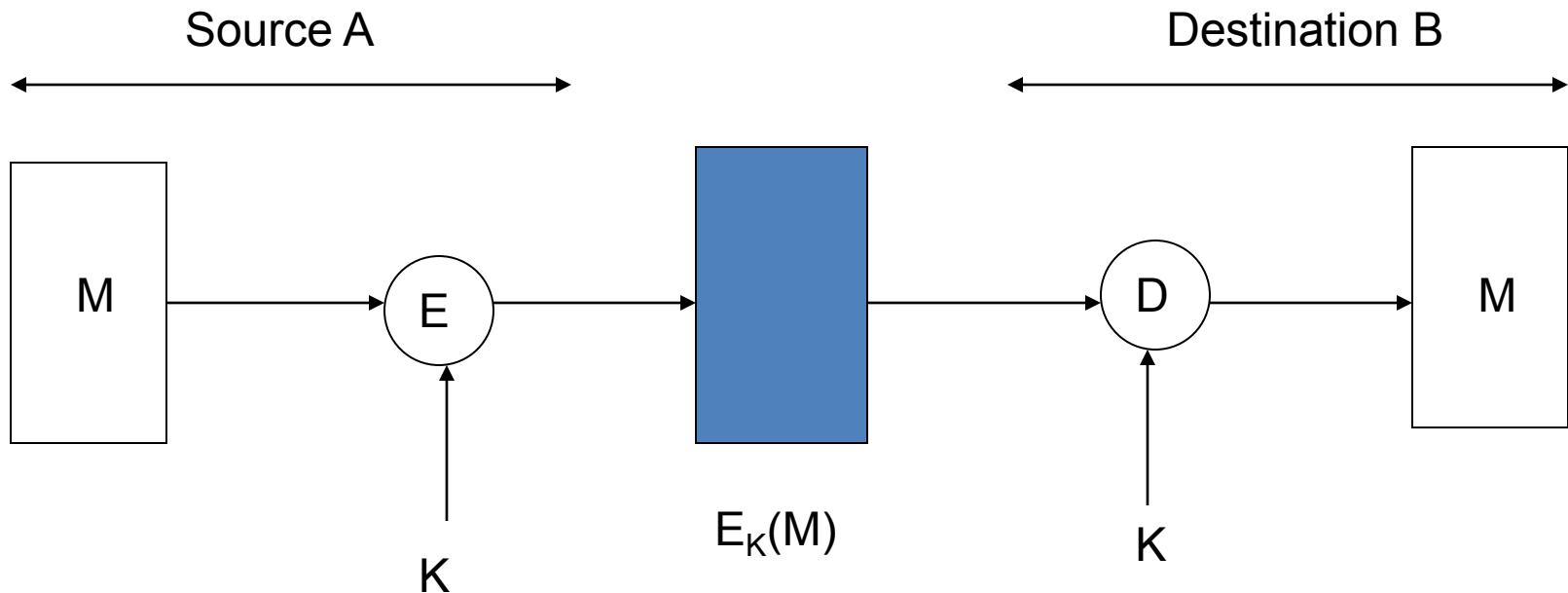
- Disclosure
- Traffic analysis
- Masquerade
- Content modification
- Sequence modification
- Timing modification
- Source repudiation
- Destination repudiation

Message Encryption

- Message encryption by itself also provides a measure of authentication
- If symmetric encryption is used then:
 - receiver knows sender must have created it since only sender and receiver know key used
 - knows content is not been altered
 - if message has suitable structure, redundancy or a checksum to detect any changes

Message Encryption

- Symmetric encryption: Confidentiality & Authentication

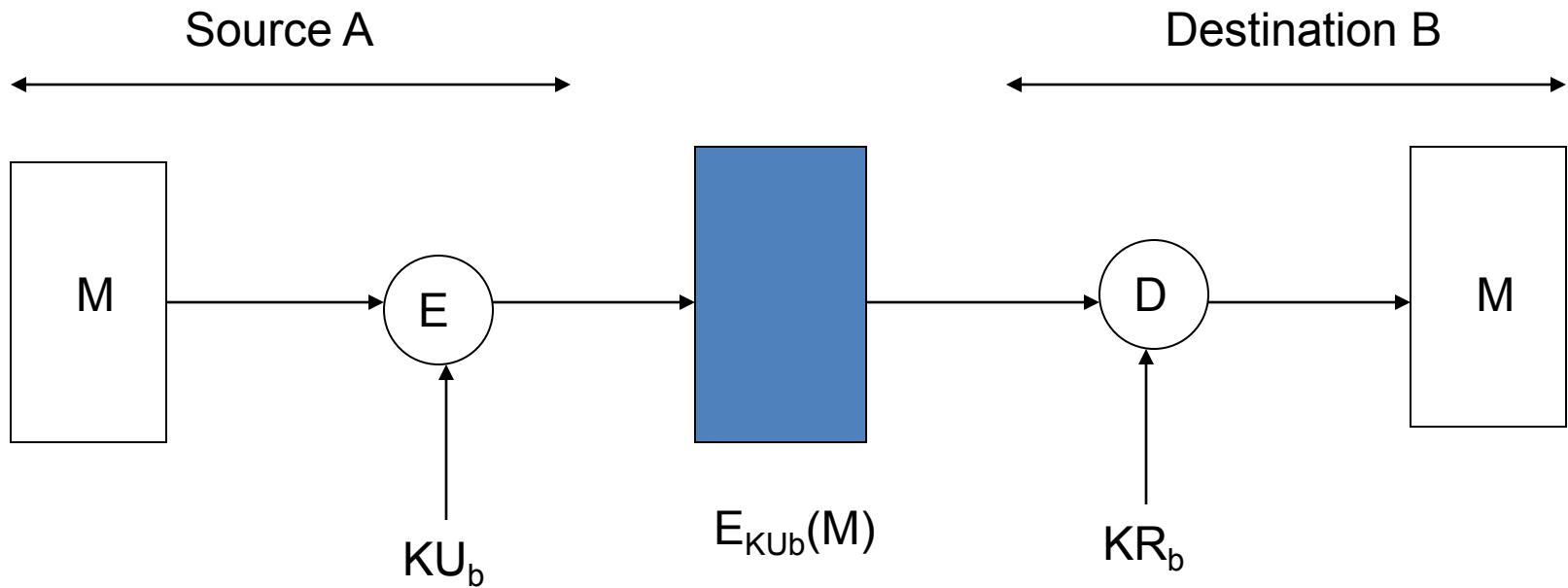


Message Encryption

- If public-key encryption is used:
 - encryption provides no confidence of sender
 - since anyone potentially knows public-key
 - however if
 - sender **signs** message using their private-key
 - then encrypts with recipients public key
 - have both secrecy and authentication

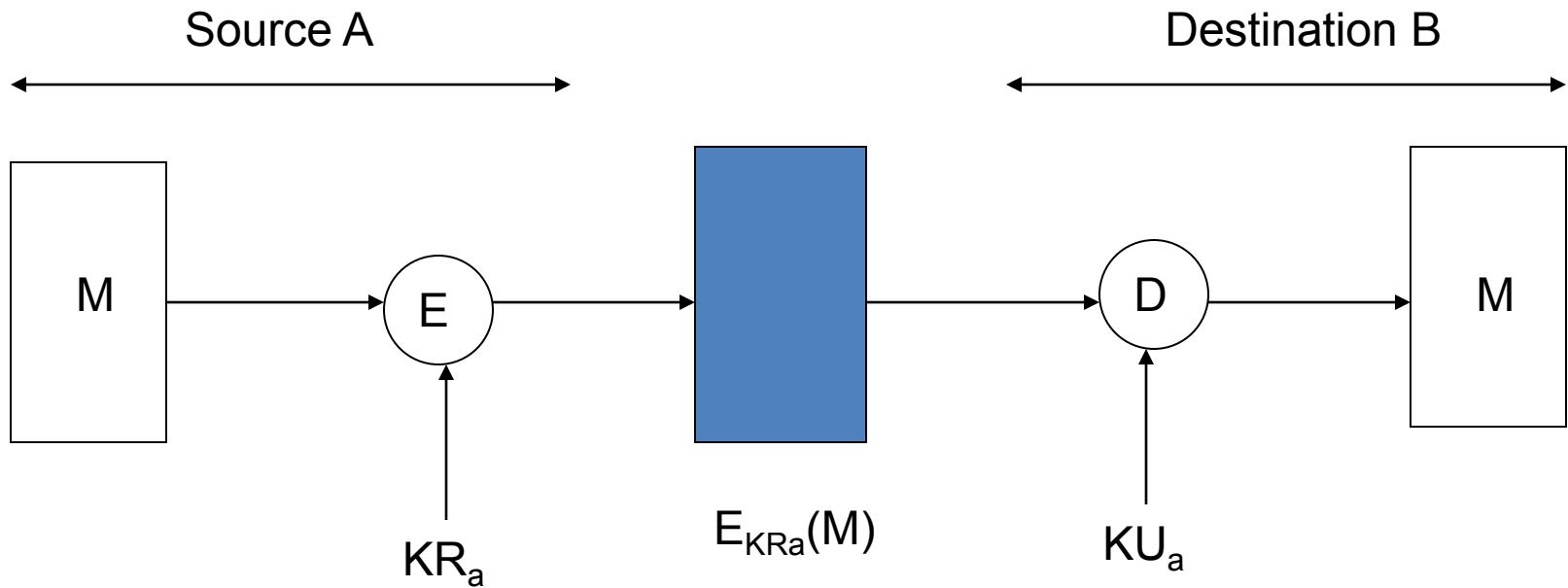
Message Encryption

- Public-key encryption : Confidentiality



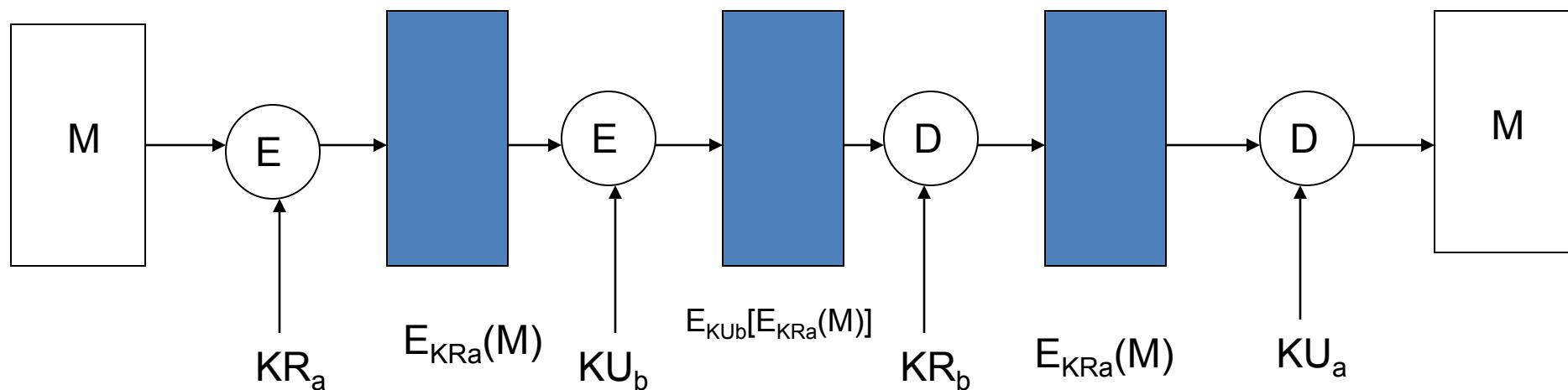
Message Encryption

- Public-key encryption : Authentication & Signature



Message Encryption

Public-key encryption : Confidentiality, Authentication & Signature



Message Authentication Code (MAC)

- Generated by an algorithm that creates a small fixed-sized block called **cryptographic checksum or MAC**
 - depending on both message and some key
 - need not be reversible like encryption
- 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

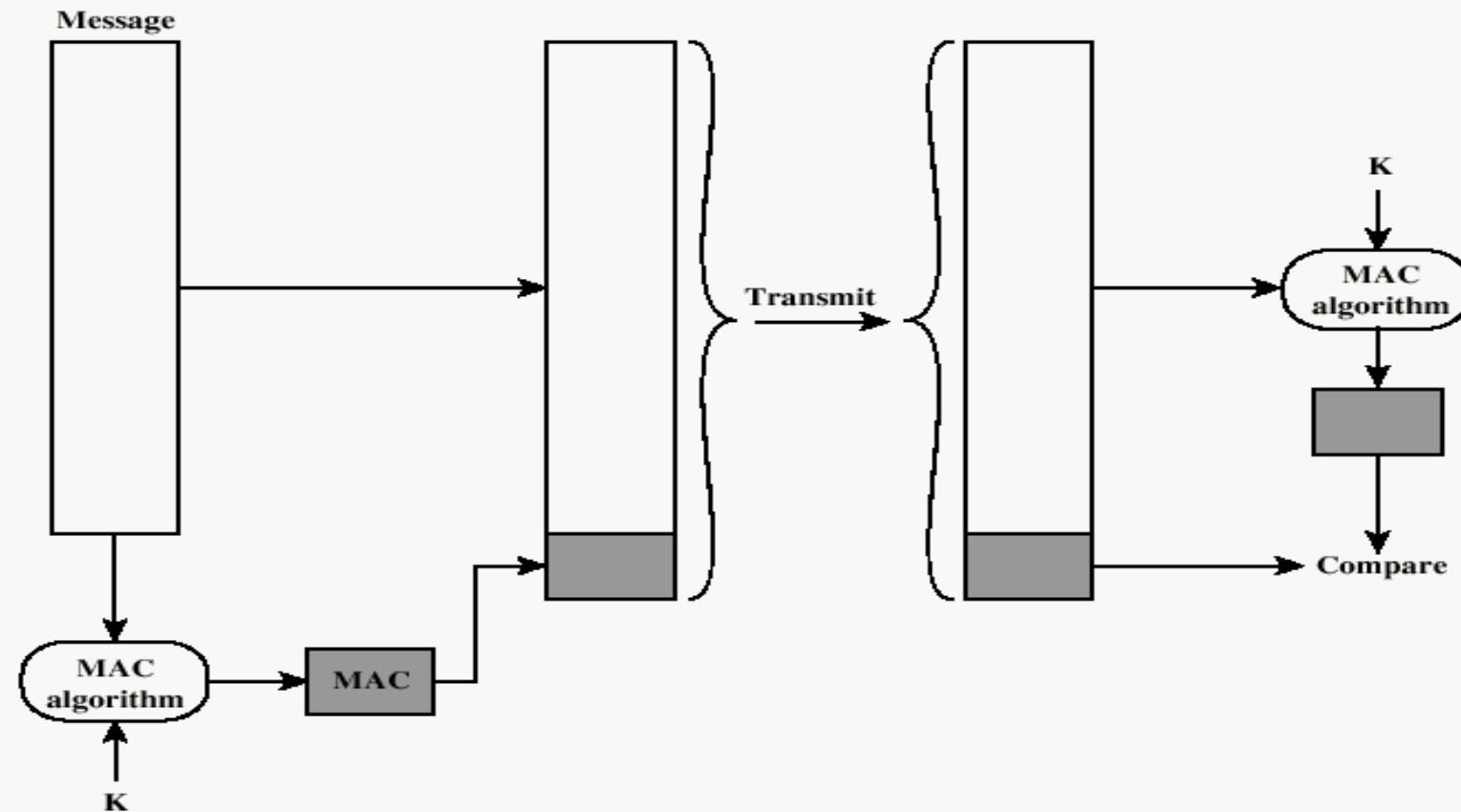


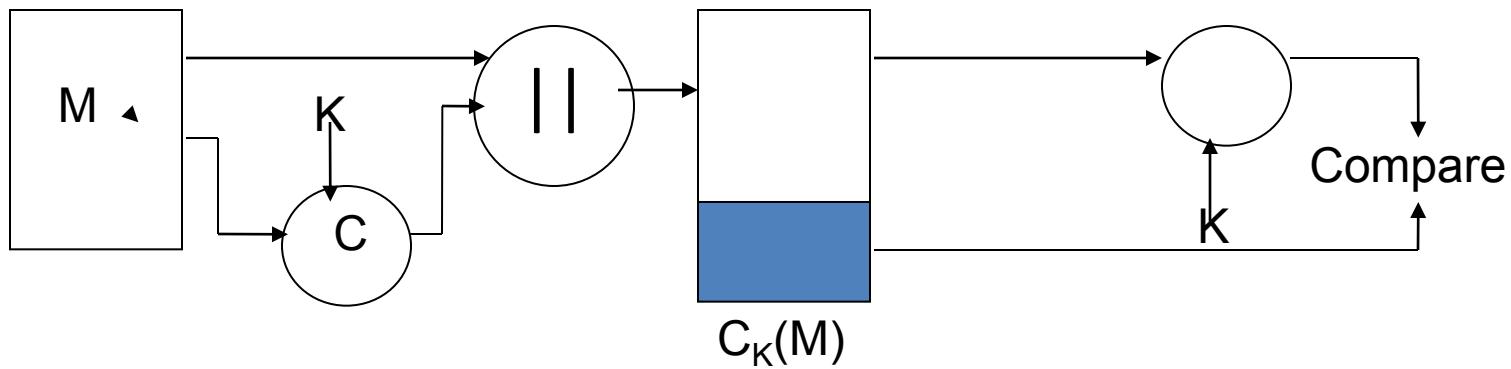
Figure 3.1 Message Authentication Using a Message Authentication Code (MAC)

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
- If only sender and receiver only knows secret key and if MAC is matched then,
 - Receiver is assured that the message has not been altered
 - Receiver is assured that the message is from the alleged sender
 - If message includes Sequence no. then the receiver can be assured of the proper sequence

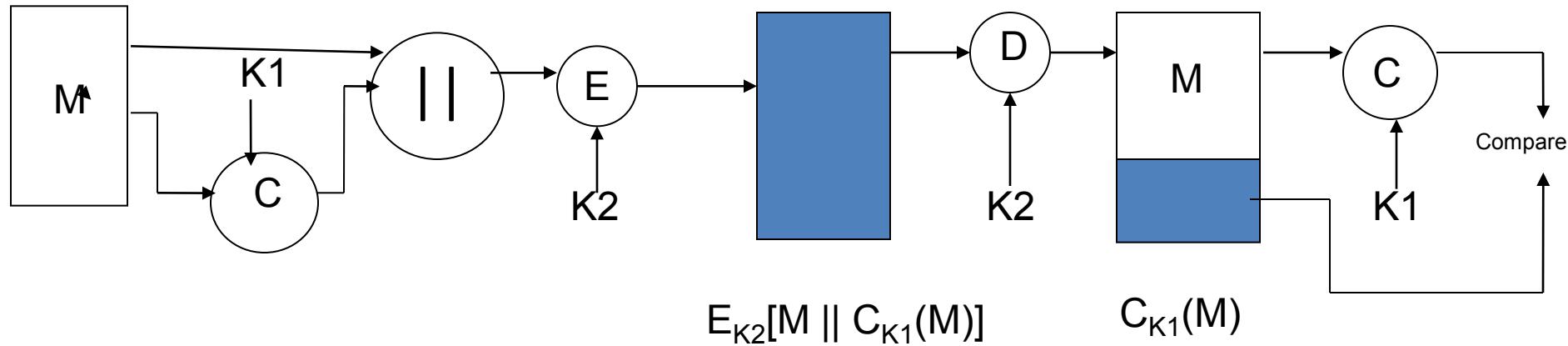
Message Authentication Codes

Message Authentication:



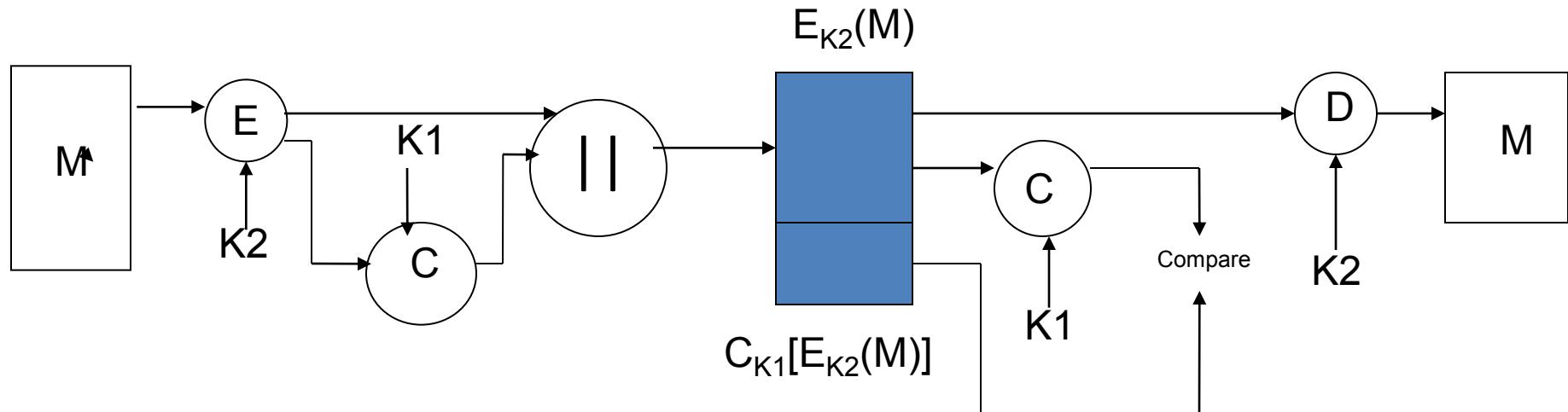
Message Authentication Codes

Message Authentication and Confidentiality: (Authentication Tied to Plaintext)



Message Authentication Codes

**Message Authentication and Confidentiality:
(Authentication Tied to Cipher text)**

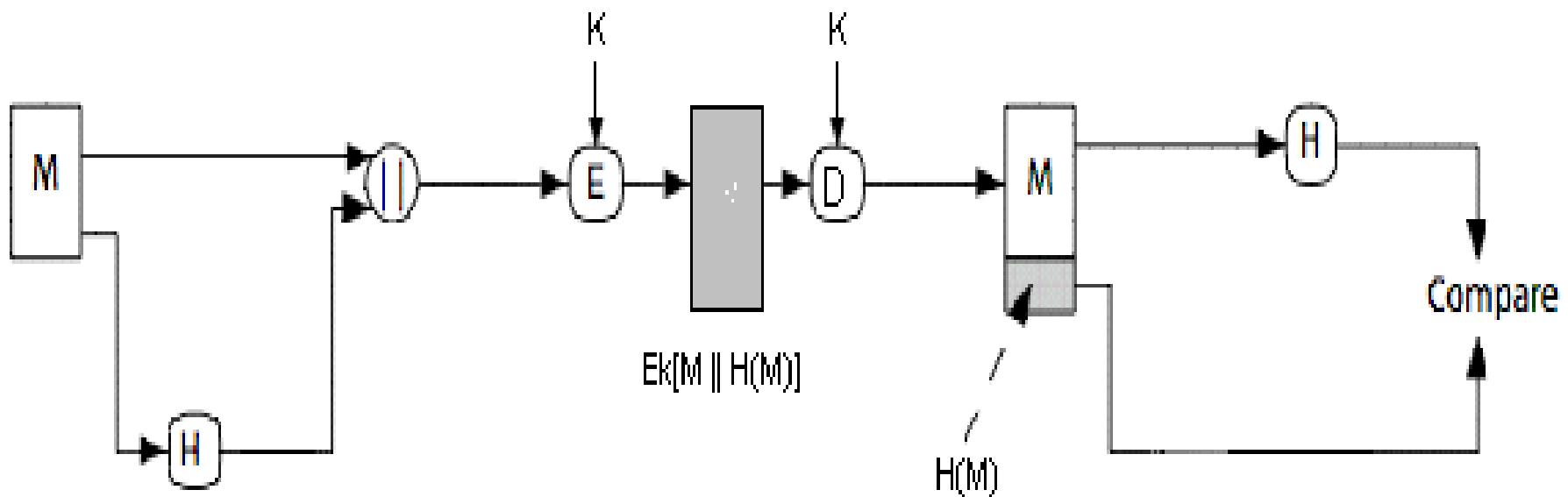


Hash Functions

- Condenses arbitrary message to fixed size Hash code $h = H(M)$
- Also called Message digest or Hash value
- The hash function is public and not keyed (MAC is keyed)
- Hash code is a function of all bits of the message
- Change to any bit or bits in the message results in a change to the Hash code
- Most often to create a digital signature
- Can use in various ways with message

Basic Uses of Hash Function

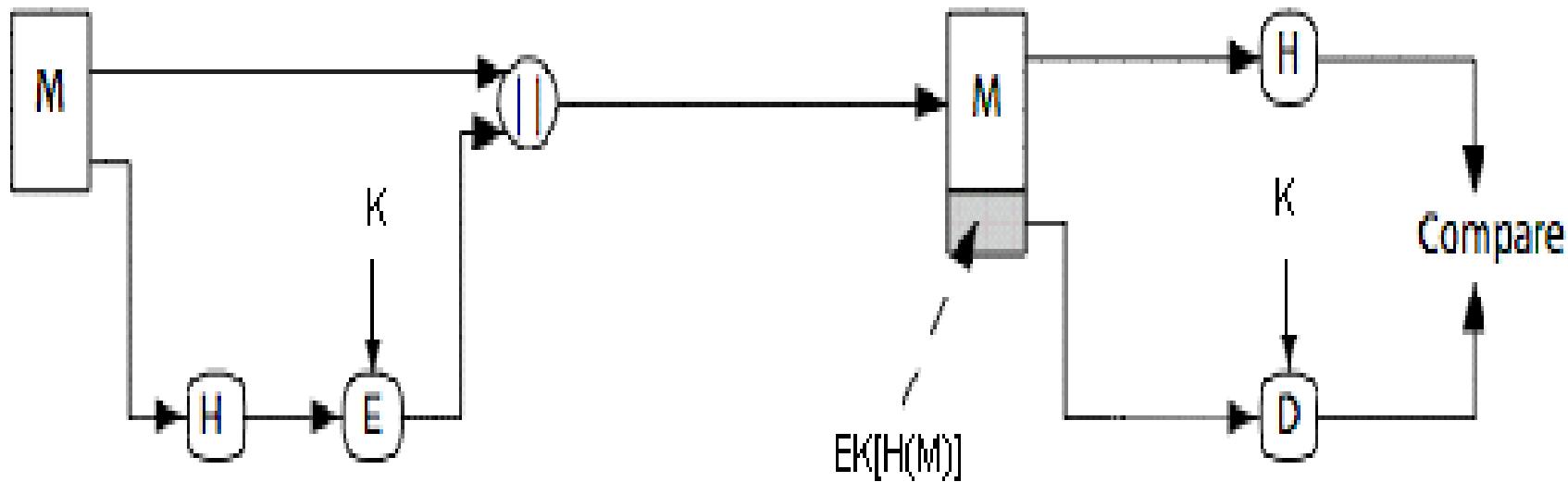
- a. Message plus concatenated hash code is encrypted using symmetric encryption



Authentication and Confidentiality

Basic Uses of Hash Function

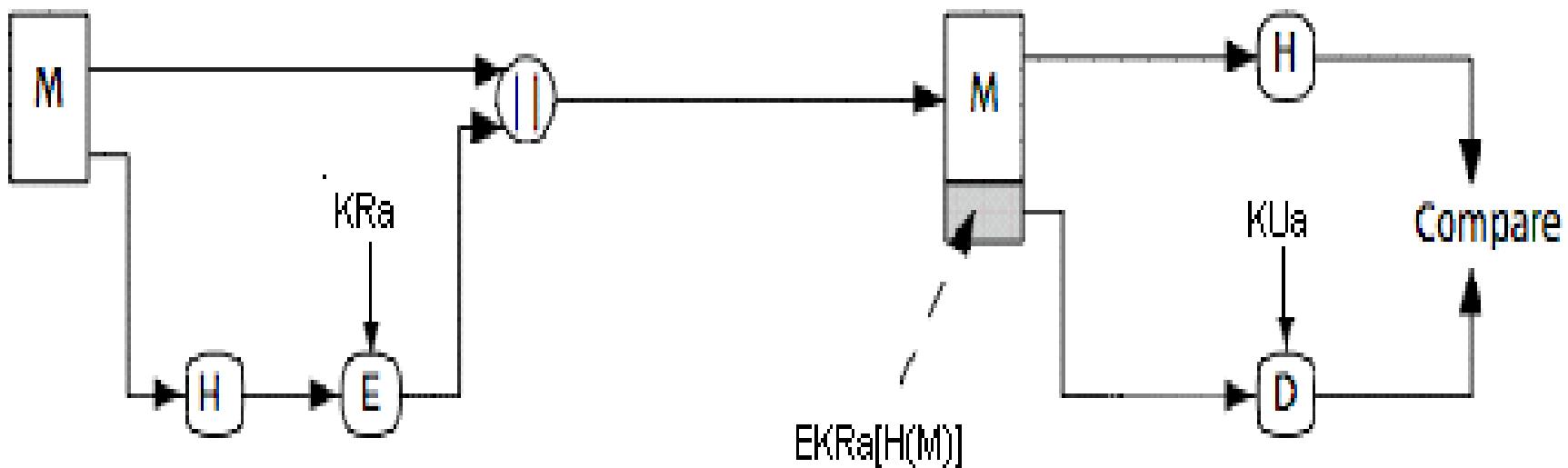
b. Only the hash code is encrypted using symmetric encryption



No Confidentiality only Authentication (Acts as MAC)

Basic Uses of Hash Function

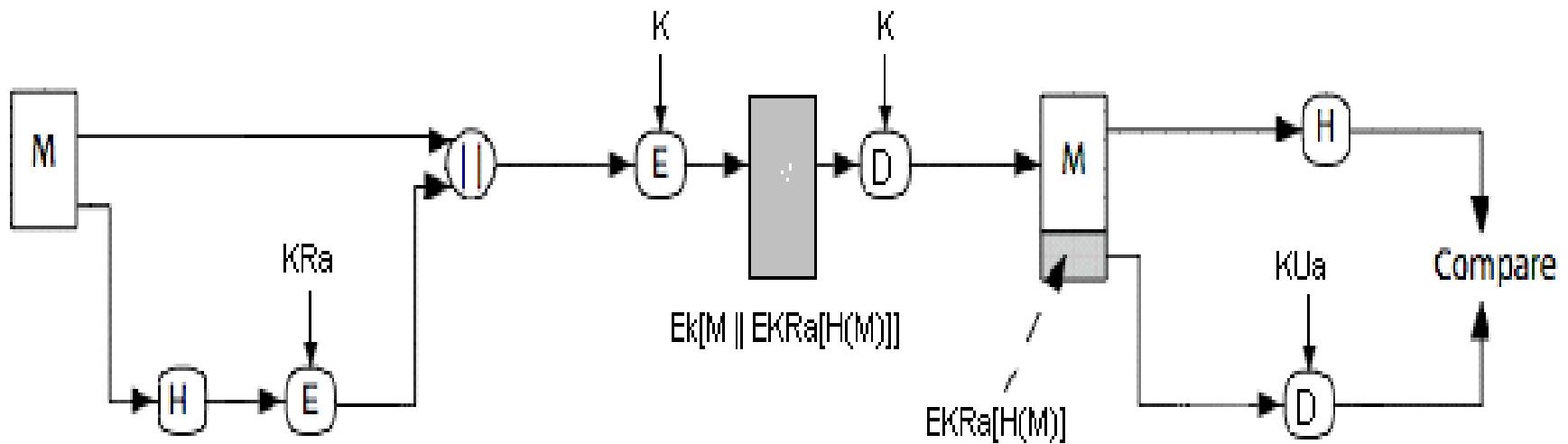
- c. Only hash code is encrypted using public key encryption using sender's private key



No Confidentiality only Authentication

Basic Uses of Hash Function

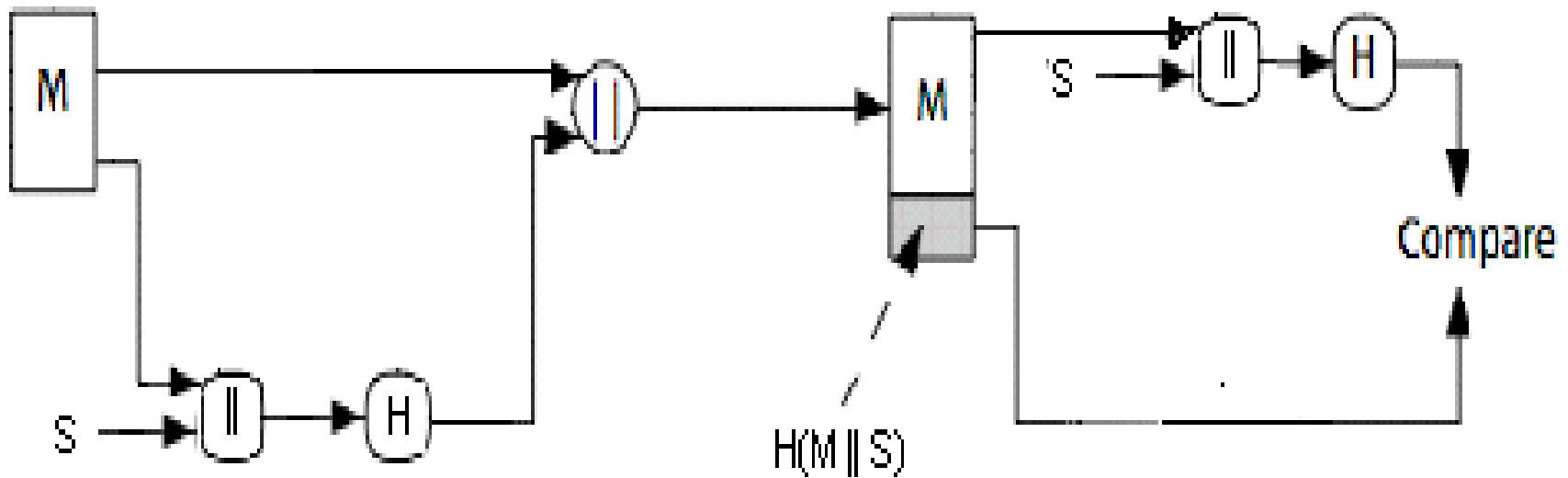
- d. Message plus public-key-encrypted hash code is encrypted using a symmetric secret key



Confidentiality and digital signature

Basic Uses of Hash Function

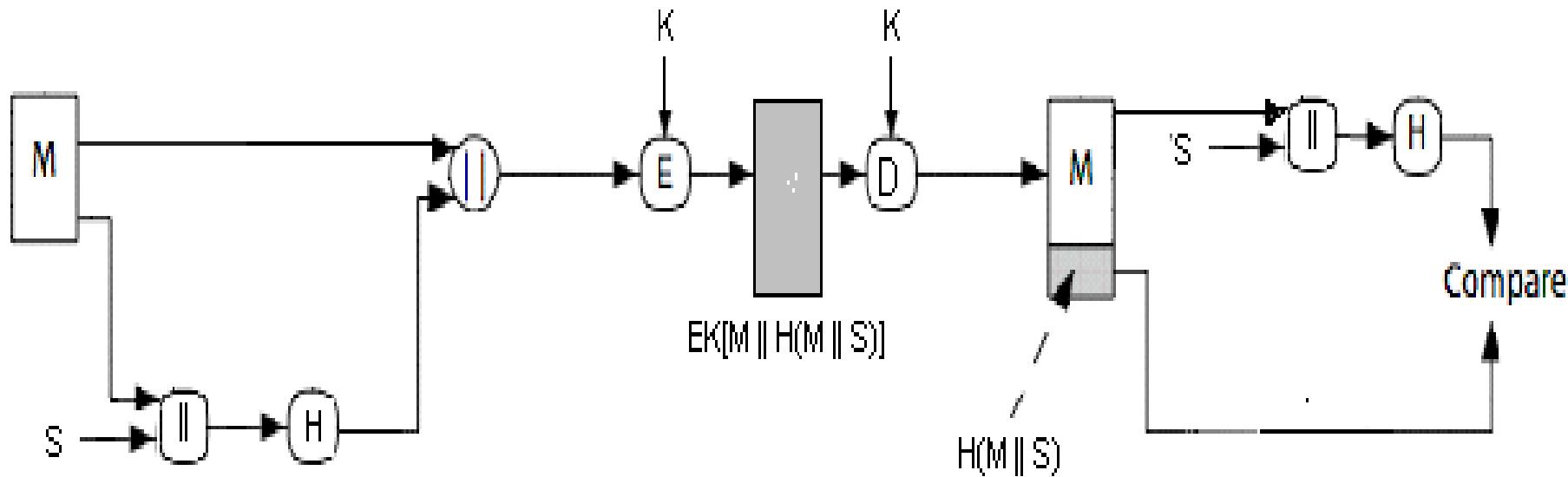
e. Hash value is computed over message plus secret value S



No Confidentiality only Authentication

Basic Uses of Hash Function

f. Message plus hash code is encrypted to the approach (e)



Confidentiality and Authentication

Requirements for Hash Functions

- The Purpose of Hash function is to produce the fingerprint of file, message or other block of data
- For message authentication, a hash function H must have the following properties:
 1. H can be applied to any sized message M
 2. H produces fixed-length output h
 3. It is easy to compute $h=H(M)$ for any message M
 4. Given h , it is infeasible to find M s.t. $H(M)=h$
 - one-way property
 5. Given x , it is infeasible to find y s.t. $H(y)=H(x)$
 - weak collision resistance
 6. It is infeasible to find any x, y s.t. $H(y)=H(x)$
 - strong collision resistance

Birthday Attacks

- Might think a 64-bit hash is secure
- But by **Birthday Paradox** is not
- The **birthday paradox** can be stated as follows:
 - What is the minimum value of k such that the probability is greater than 0.5 that at least two people in a group of k people have the same birthday?
 - It turns out that the answer is 23 which is quite a surprising result.
 - In other words if there are 23 people in a room, the probability that two of them have the same birthday is approximately 0.5.
 - If there is 100 people (i.e. $k=100$) then the probability is .9999997, i.e. you are almost guaranteed that there will be a duplicate.

Birthday Attacks

- Digital signatures can be susceptible to a birthday attack.
- A message is typically signed by first computing , where is a cryptographic hash function, and then using some secret key to sign .
- Suppose Mallory wants to trick Bob into signing a fraudulent contract.
- Mallory prepares a fair contract and a fraudulent one .
- She then finds a number of positions where can be changed without changing the meaning, such as inserting commas, empty lines, one versus two spaces after a sentence, replacing synonyms, etc.

Birthday Attacks

- By combining these changes, she can create a huge number of variations on which are all fair contracts.
- In a similar manner, Mallory also creates a huge number of variations on the fraudulent contract .
- She then applies the hash function to all these variations until she finds a version of the fair contract and a version of the fraudulent contract which have the same hash value, .
- She presents the fair version to Bob for signing.
- After Bob has signed, Mallory takes the signature and attaches it to the fraudulent contract.
- This signature then "proves" that Bob signed the fraudulent contract.

Birthday Attacks

- Might think a 64-bit hash is secure
- But by **Birthday Paradox** is not
- **Birthday attack** works thus:
 - opponent generates $2^{m/2}$ variations of a valid message all with essentially the same meaning
 - opponent also generates $2^{m/2}$ variations of a desired fraudulent message
 - two sets of messages are compared to find pair with same hash (probability > 0.5 by birthday paradox)
 - user sign the valid message, then substitute the forgery which will have a valid signature
- Conclusion is that need to use larger MAC/hash

MD5: Message Digest Version 5

Input Message

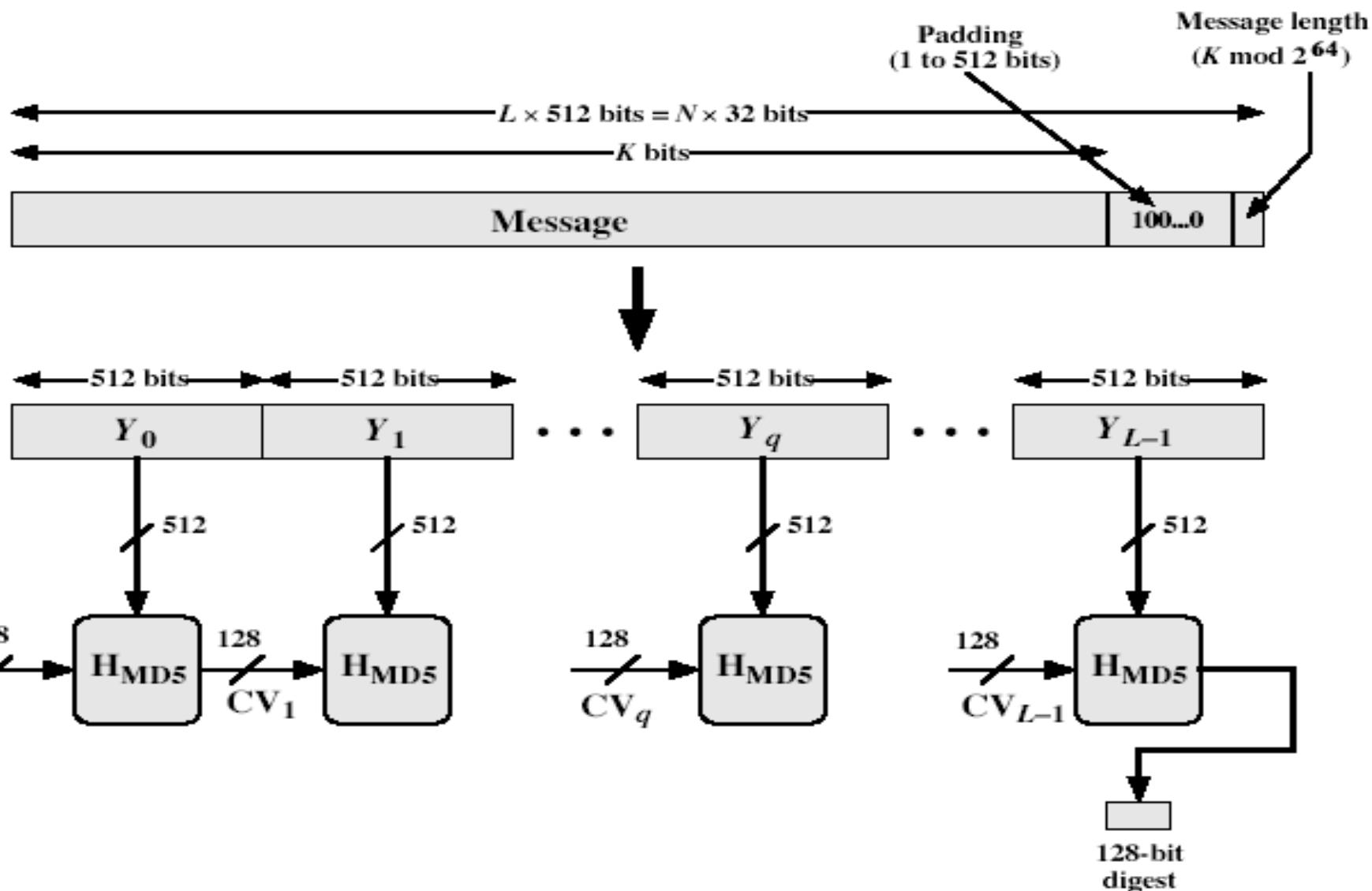


Output 128 bits Digest

- Developed by Ron Rivest at MIT
- Until recently the most widely used hash algorithm
- Specified as Internet standard RFC1321

www.rejinpaul.com

MD5 Overview



MD5 Overview

1. Append Padding Bits:

- Pad message so its length is $448 \bmod 512$
- ie the length of padded message is 64 bits less than an integer multiple of 512 bits.
- Padding is always added
- For eg. If the message is 448 bits long, it is padded by 512 bits to a length of 960 bits
- The number of padding bits is in the range of 1-512
- Padding consists of single 1-bit followed by the necessary no of 0-bits

MD5 Overview

2. Append Length:

- A 64-bit length value of original message is appended to the result of step 1.
 - If the original message is greater than 2^{64} then only the lower order 64 bits of the length are used.
 - Thus the field contains length of the original message
-
- The outcome of the first 2 steps yields the message that is an integer multiple of 512 bits in length.
 - Expanded message is represented as the sequence of 512-bit blocks $Y_0, Y_1 \dots Y_{L-1}$
 - Total length of the expanded message is $L \times 512$ bits

MD5 Overview

3. Initialize MD buffer:

- A 128-bit buffer is used to hold the intermediate and final results of the hash function.
- Buffer can be represented as four 32-bit registers (A,B,C,D)
- These registers are initialized to the following 32-bit integers:

A = 67 45 23 01

B = EF CD AB 89

C = 98 BA DC FE

D = 10 32 54 76

- These values are stored in little-endian format

word A = 01 23 45 67

word B = 89 AB CD EF

word C = FE DC BA 98

word D = 76 54 32 10

MD5 Overview

4. Process message in 512-bit (16-word) blocks: (Compression Function)

- Using 4 rounds of 16 bit operations on message block & buffer
- These 4 rounds have similar structures but uses different logical functions, F, G, H and I
- Each round takes the current 512-bit block (Y_q) and 128-bit buffer value ABCD as input and updates the contents of the buffer.
- Each round also makes use of the one fourth of a 64-element table $T[1\dots64]$ constructed from sine function.
- The i^{th} element of T , $T[i] = 2^{32} \times \text{abs}(\sin(i))$ (i is in radians)
- The output of the fourth round is added to the input to the first round (CV_q) to produce CV_{q+1} .
- This addition is done using addition modulo 2^{32} .

MD5 Overview

5. Output:

- After all L 512-bit blocks have been processed, the output from the L^{th} stage is the 128-bit message digest.
- Behavior of MD5 can be summarized:

$$CV_0 = IV$$

$$CV_{q+1} = \text{SUM}_{32}[CV_q, RF_I(Y_q, RF_H(Y_q, RF_G(Y_q, RF_F(Y_q, CV_q))))]$$

where

IV = Initial value of ABCD buffer

Y_q = qth 512-bit block of the message

L = No. of blocks in the message

CV_q = Chaining Variable processed with qth block

RF_x = Round Function using primitive legal fn x

MD = Final message digest

SUM_{32} = Addition modulo 2^{32}

MD5 Compression Function

Each round has 16 steps of the form:

$$b = b + (a + g(b, c, d) + X[k] + T[i]) \ll s$$

Where

a, b, c, d = the 4 words of the buffer

g = one of the primitive fn F, G, H, I

$\ll s$ = circular left shift of the 32-bit argument by s Bits

$X[k] = M[q \times 16 + k]$ = kth 32-bit word in the qth 512-bit block of the message

$X[i]$ = In the first round – used in their original order

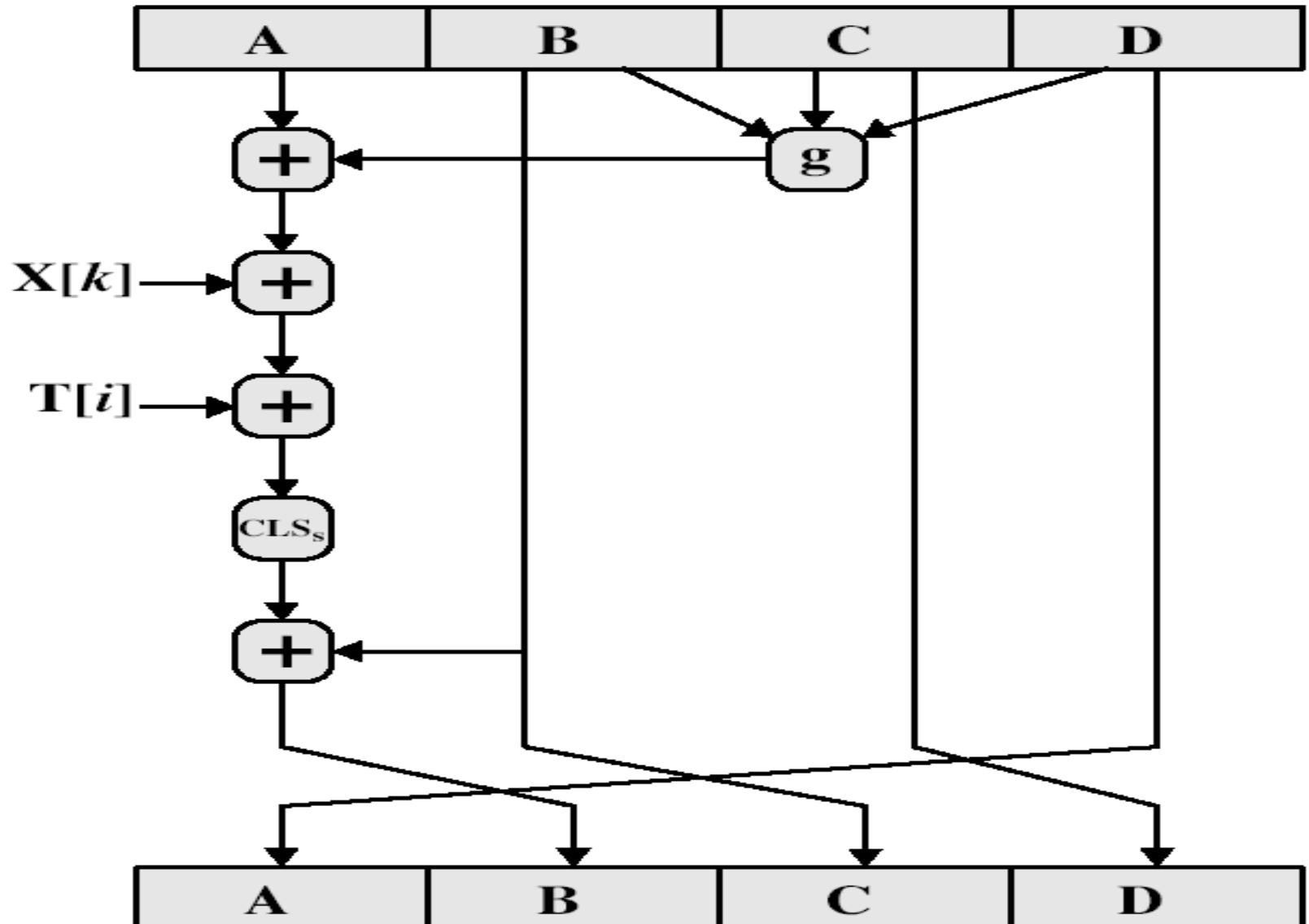
$$p2[i] = (1 + 5i) \bmod 16$$

$$p3[i] = (5 + 3i) \bmod 16$$

$$p4[i] = 7i \bmod 16$$

$T[i]$ = ith 32-bit word in matrix T += Addition modulo 2^{32}

MD5 Compression Function



Functions F, G, H and I

Round	Primitive Fn. g	$G(b, c, d)$
1	$F(b,c,d)$	$(b \wedge c) \vee (\neg b \wedge d)$
2	$G(b,c,d)$	$(b \wedge d) \vee (c \wedge \neg d)$
3	$H(b,c,d)$	$b \oplus c \oplus d$
4	$I(b,c,d)$	$c \oplus (b \wedge \neg d)$

Truth Table of Logical Functions

b	c	d	F	G	H	I
0	0	0	0	0	0	1
0	0	1	1	0	1	0
0	1	0	0	1	1	0
0	1	1	1	0	0	1
1	0	0	0	0	1	1
1	0	1	0	1	0	1
1	1	0	1	1	0	0
1	1	1	1	1	1	0

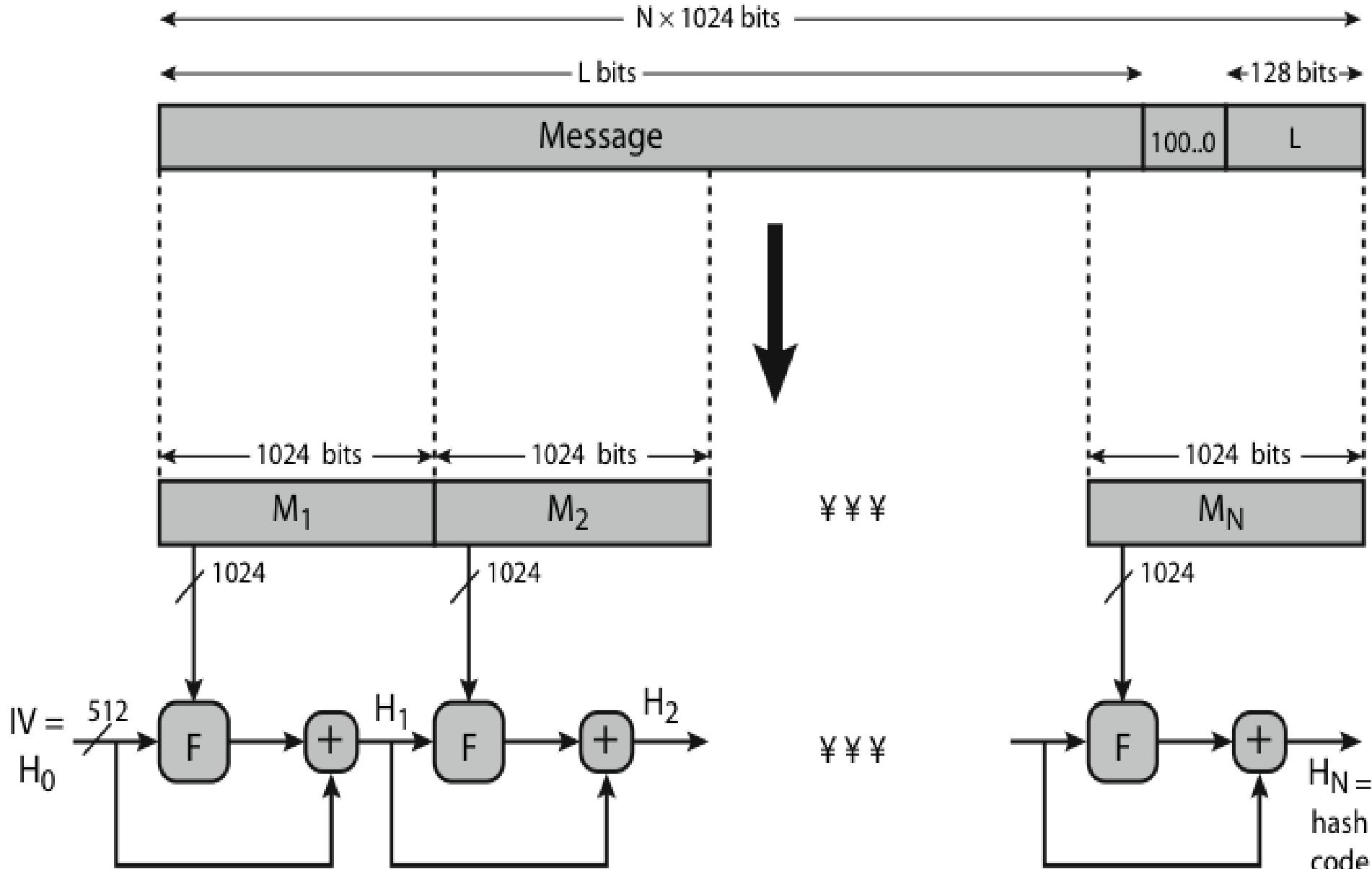
Secure Hash Algorithm

- SHA originally designed by NIST & NSA in 1993 and was revised in 1995 as SHA-1
- US standard for use with DSA signature scheme
 - standard is FIPS 180-1 1995, also Internet RFC3174
 - nb. the algorithm is SHA, the standard is SHS
- Based on design of MD4 with key differences
- Produces 160-bit hash values
- Recent 2005 results on security of SHA-1 have raised concerns on its use in future applications

Revised Secure Hash Standard

- NIST issued revision FIPS 180-2 in 2002 and added 3 additional versions of SHA
 - SHA-256, SHA-384, SHA-512
- Designed for compatibility with increased security provided by the AES cipher
- Structure & detail is similar to SHA-1
- Hence analysis should be similar
- But security levels are rather higher

SHA-512 Overview



SHA-512 Overview

1. Append Padding Bits:

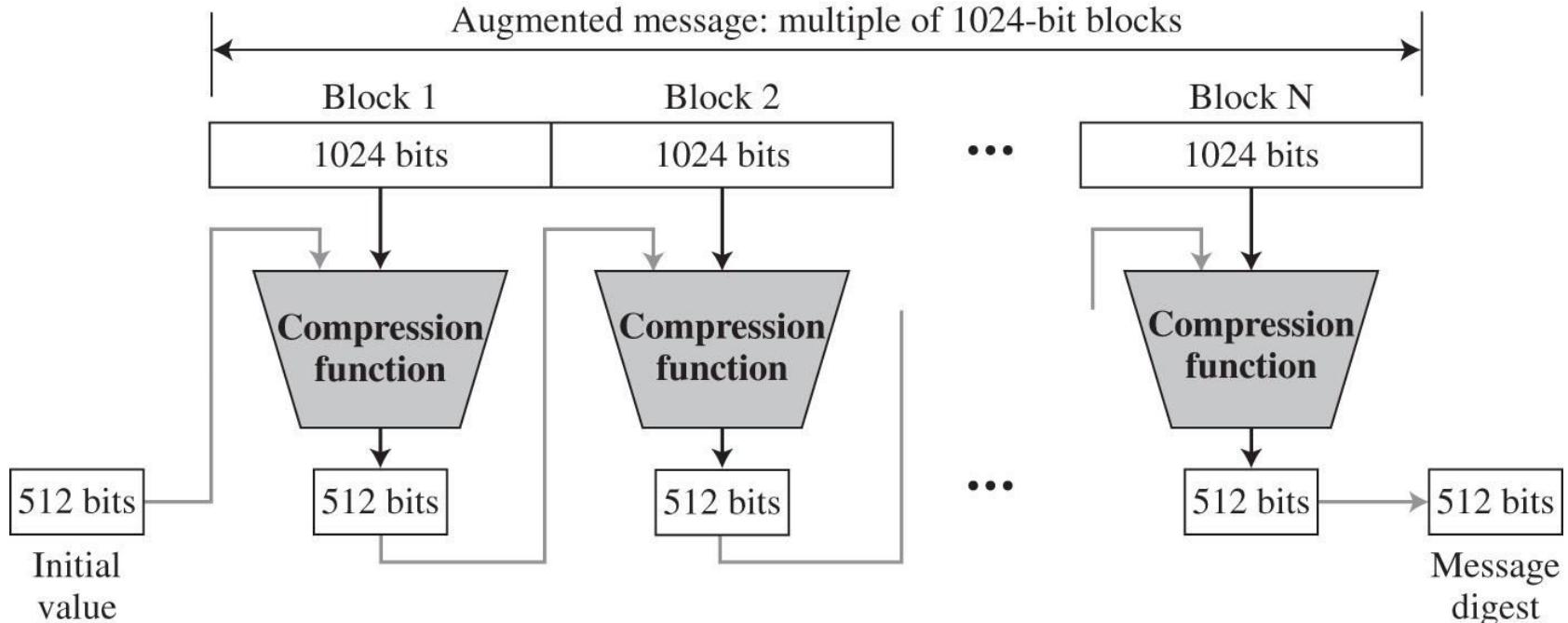
- Pad message so its length is congruent to 1024
- Padding is always added
- Padding consists of single 1-bit followed by the necessary no of 0-bits

2. Append Length:

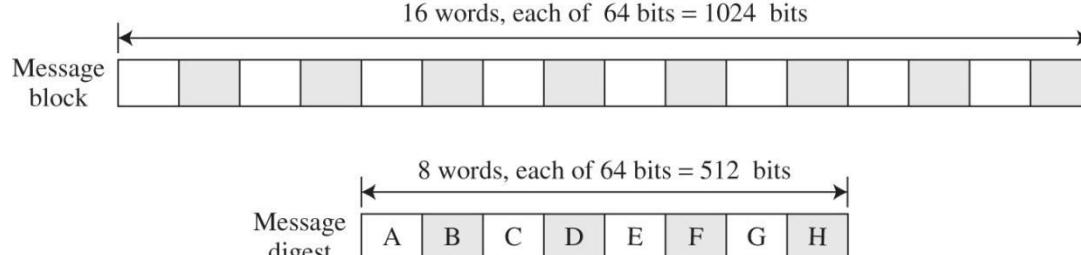
- A 128-bit length value of original message is appended to the result of step 1.
- If the original message is greater than 2^{128} then only the lower order 128 bits of the length are used.
- Thus the field contains length of the original message

SHA-512

- **512 bit message digest (secure against brute force attack)**
 - Block size: **1024 bits**



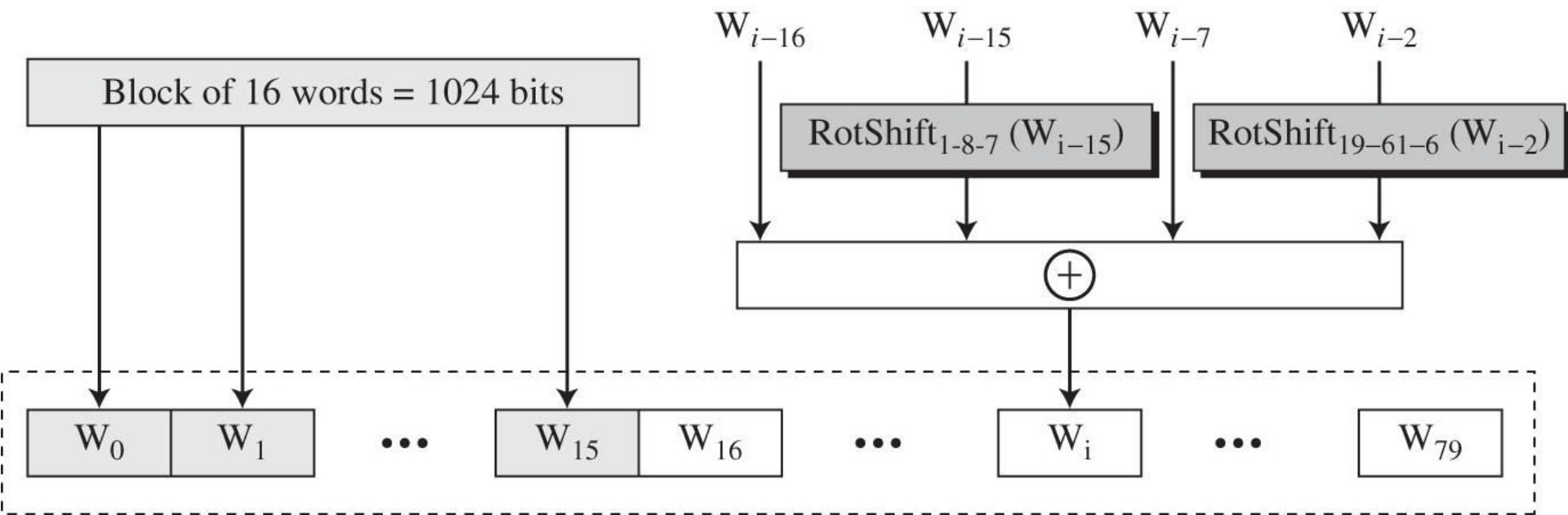
- Digest broken down into 64 bit “words” called **A – H**



Word Expansion in SHA-512

3. Word Expansion:

- Block of 16 words expanded to 80 words
 - Used by 80-round compression function

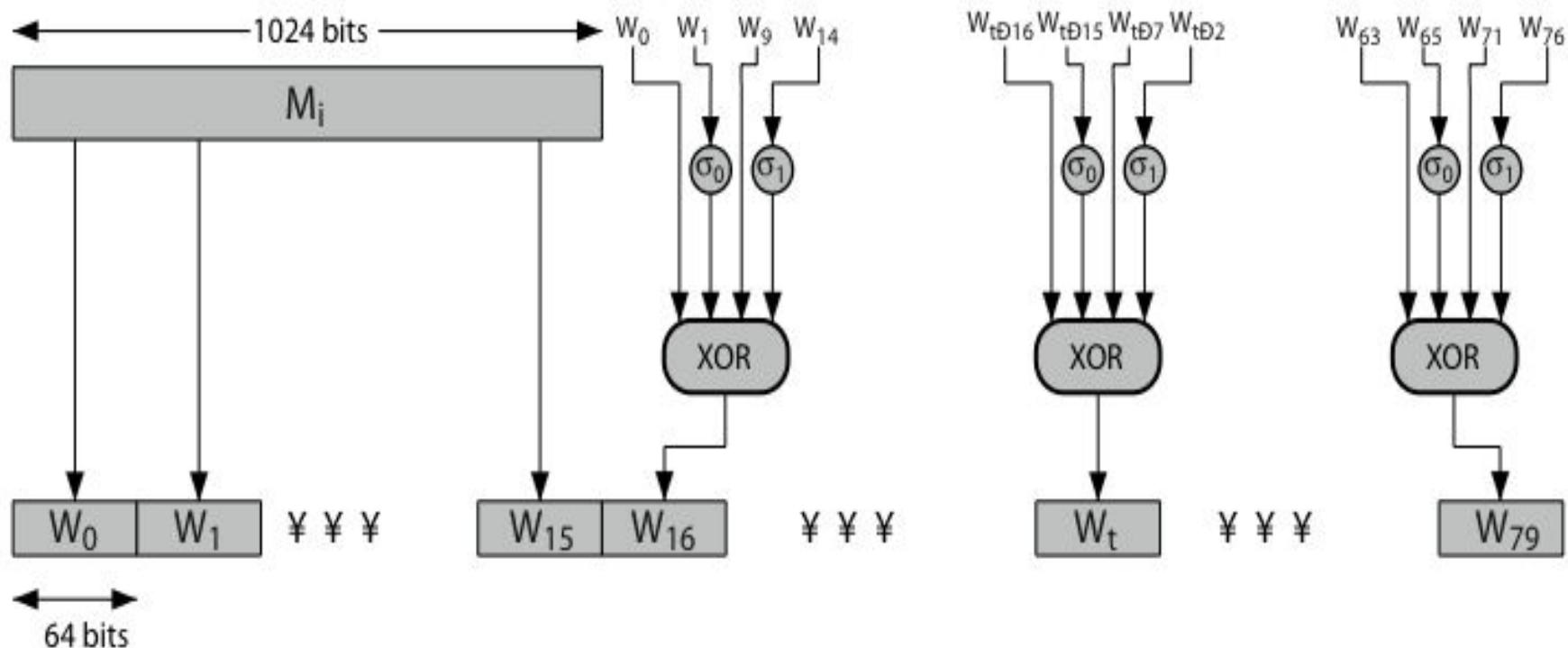


$$\text{RotShift}_{1-m-n}(x) : \text{RotR}_l(x) \oplus \text{RotR}_m(x) \oplus \text{ShL}_n(x)$$

$\text{RotR}_i(x)$: Right-rotation of the argument x by i bits

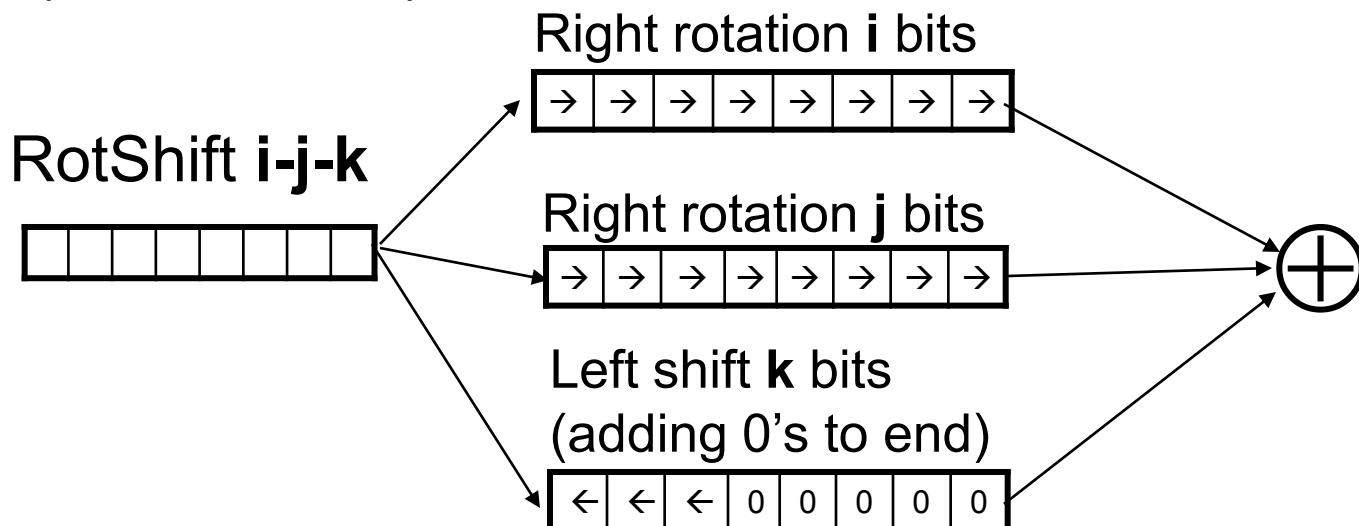
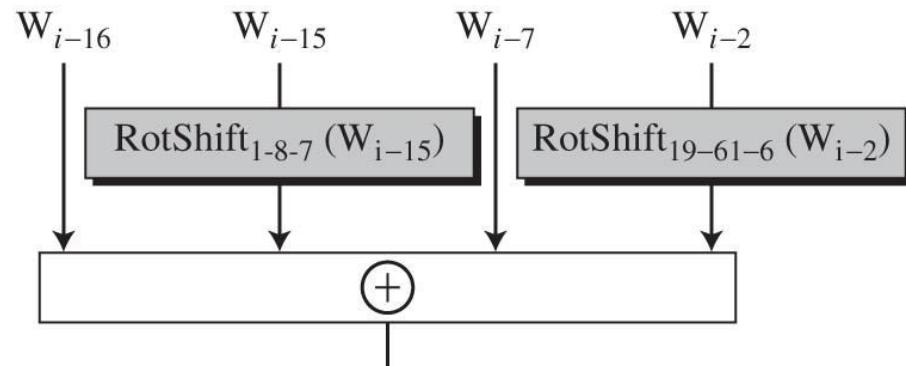
$\text{ShL}_i(x)$: Shift-left of the argument x by i bits and padding the left by 0's.

SHA-512 Round Function



Word Expansion in SHA-512

- Each word function of previous 4 words
 - Combined with XOR
 - Confusion added with rotation and shifting (not invertible)



SHA-512 Initial Digest

5. Initializing values of Buffers

- Designed for appearance of “randomness”
 - Created from first 8 primes (2, 3, 5, 7, 11, 13, 17, 19)
 - Take square root
 - Take first 64 digits of fractional part

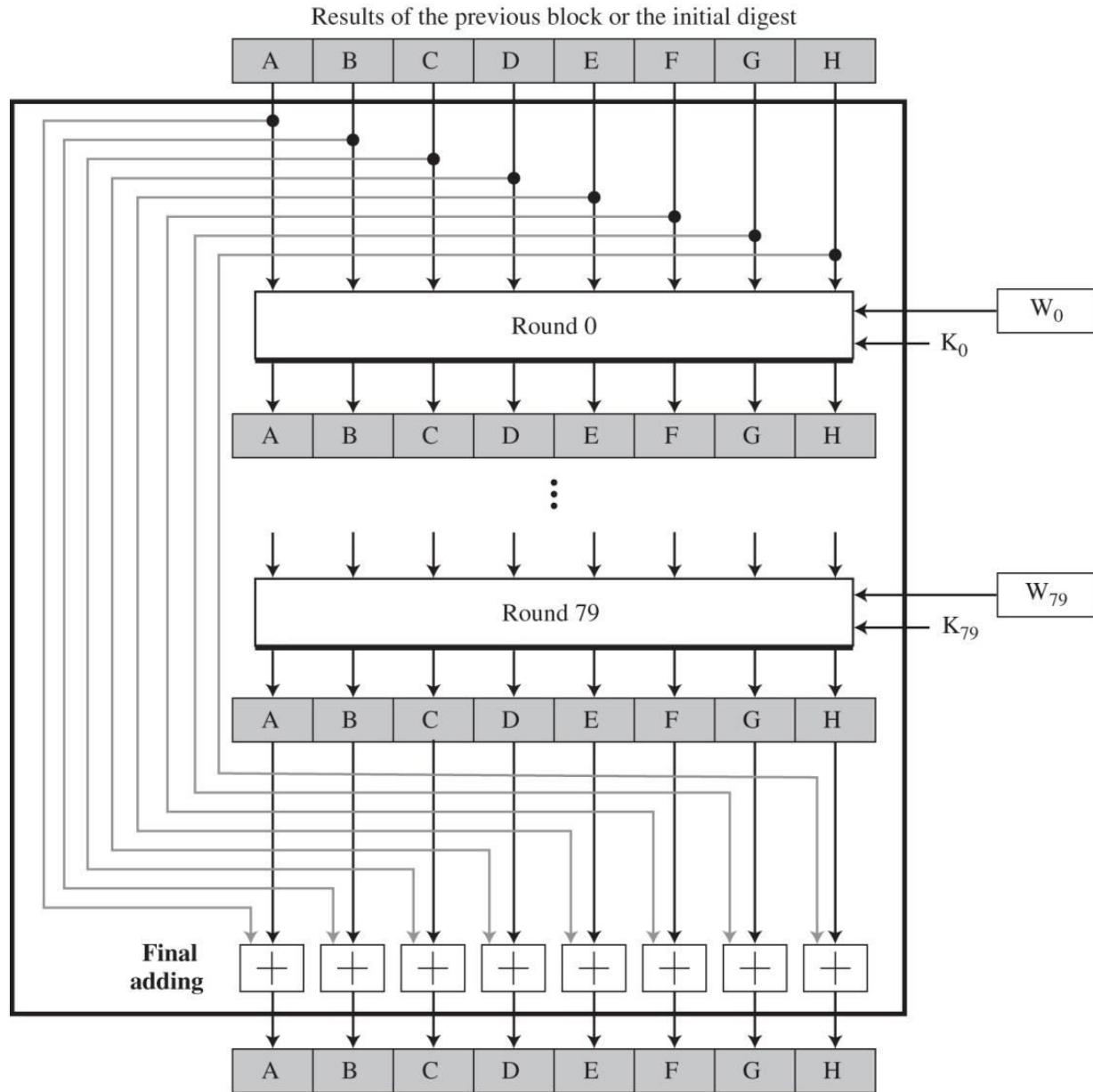
Message digest

8 words, each of 64 bits = 512 bits

A	6A09E667F3BCC908
B	BB67AE8584CAA73B
C	3C6EF372EF94F828
D	A54FE53A5F1D36F1
E	510E527FADE682D1
F	9B05688C2B3E6C1F
G	1F83D9ABFB41BD6B
H	5BE0CD19137E2179

SHA-512 Compression Function

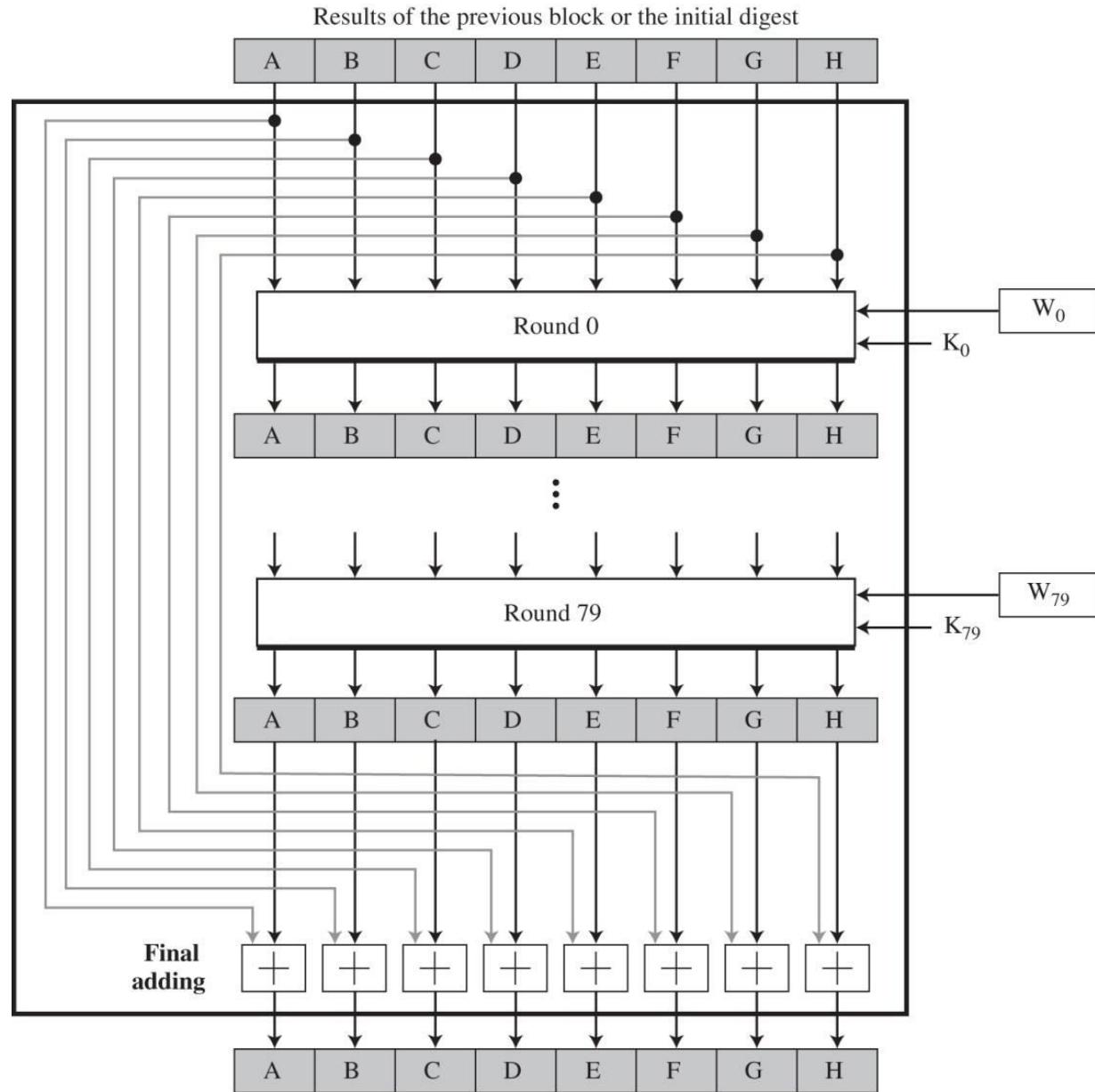
- 80 rounds
 - Each creates new “intermediate” message digest
- Final stage is sum $(\text{mod } 2^{64})$ of:
 - Initial round digest
 - Final round digest



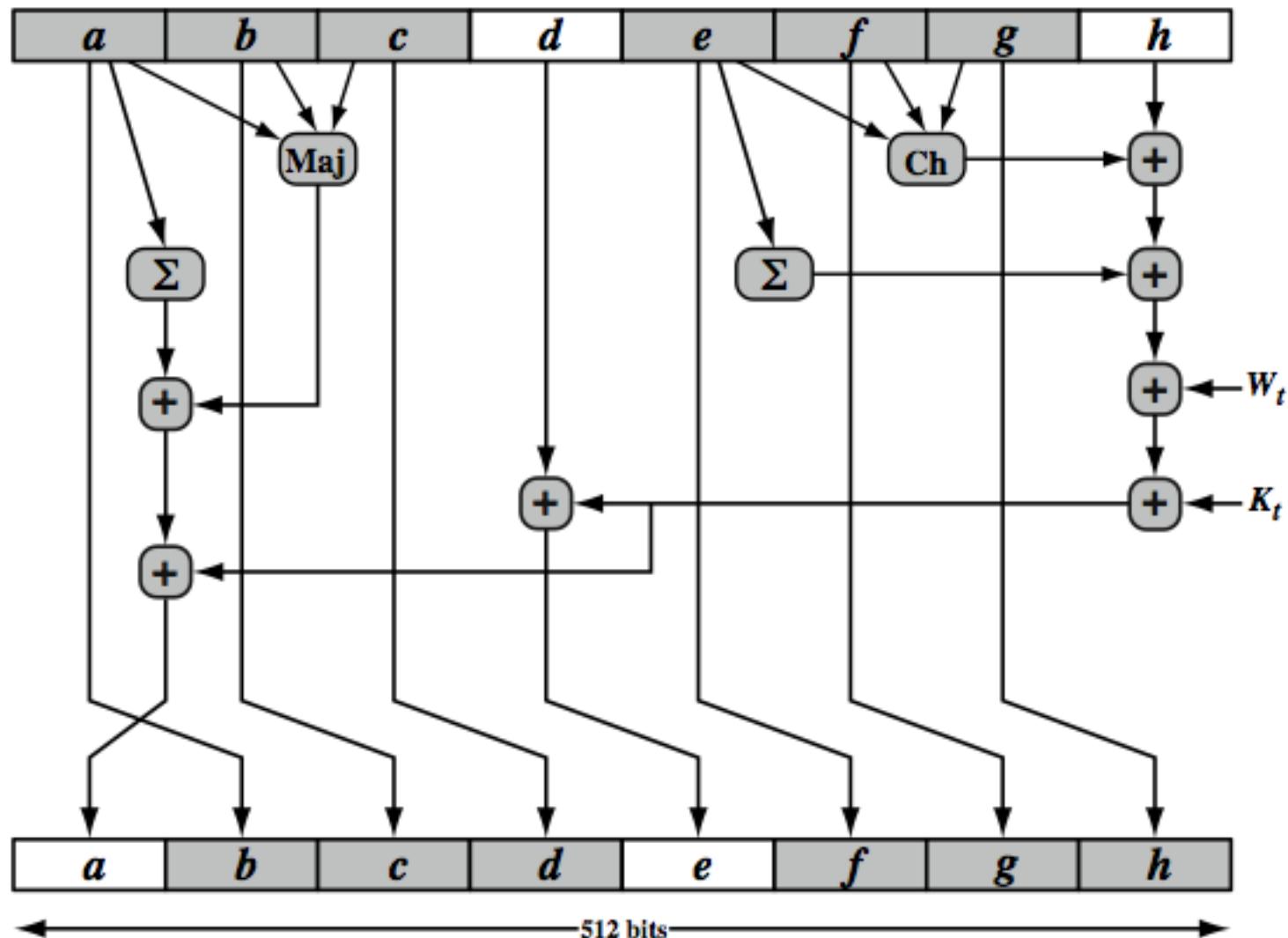
SHA-512 Compression Function

Each round i
function of:

- Previous message digest
- Word W_i
- Round “key” K_i , created from fractional parts of square root of first 80 prime numbers (like initial message digest values)



SHA-512 Round Function



SHA-512 Round Function

$$Ch(e,f,g) = (e \text{ AND } f) \text{ XOR } (\text{NOT } e \text{ AND } g)$$

$$Maj(a,b,c) = (a \text{ AND } b) \text{ XOR } (a \text{ AND } c) \text{ XOR } (b \text{ AND } c)$$

$$\Sigma(a) = \text{ROTR}(a,28) \text{ XOR } \text{ROTR}(a,34) \text{ XOR } \text{ROTR}(a,39)$$

$$\Sigma(e) = \text{ROTR}(e,14) \text{ XOR } \text{ROTR}(e,18) \text{ XOR } \text{ROTR}(e,41)$$

+ = addition modulo 2^{64}

Kt = a 64-bit additive constant

Wt = a 64-bit word derived from the current 512-bit input block.

SHA-512 Round Function

$$T_1 = h + \text{Ch}(e, f, g) + \left(\sum_1^{512} e \right) + W_t + K_t$$

$$T_2 = \left(\sum_0^{512} a \right) + \text{Maj}(a, b, c)$$

$$a = T_1 + T_2$$

$$b = a$$

$$c = b$$

$$d = c$$

$$e = d + T_1$$

$$f = e$$

$$g = f$$

$$h = g$$

Dr. B. Shanthini

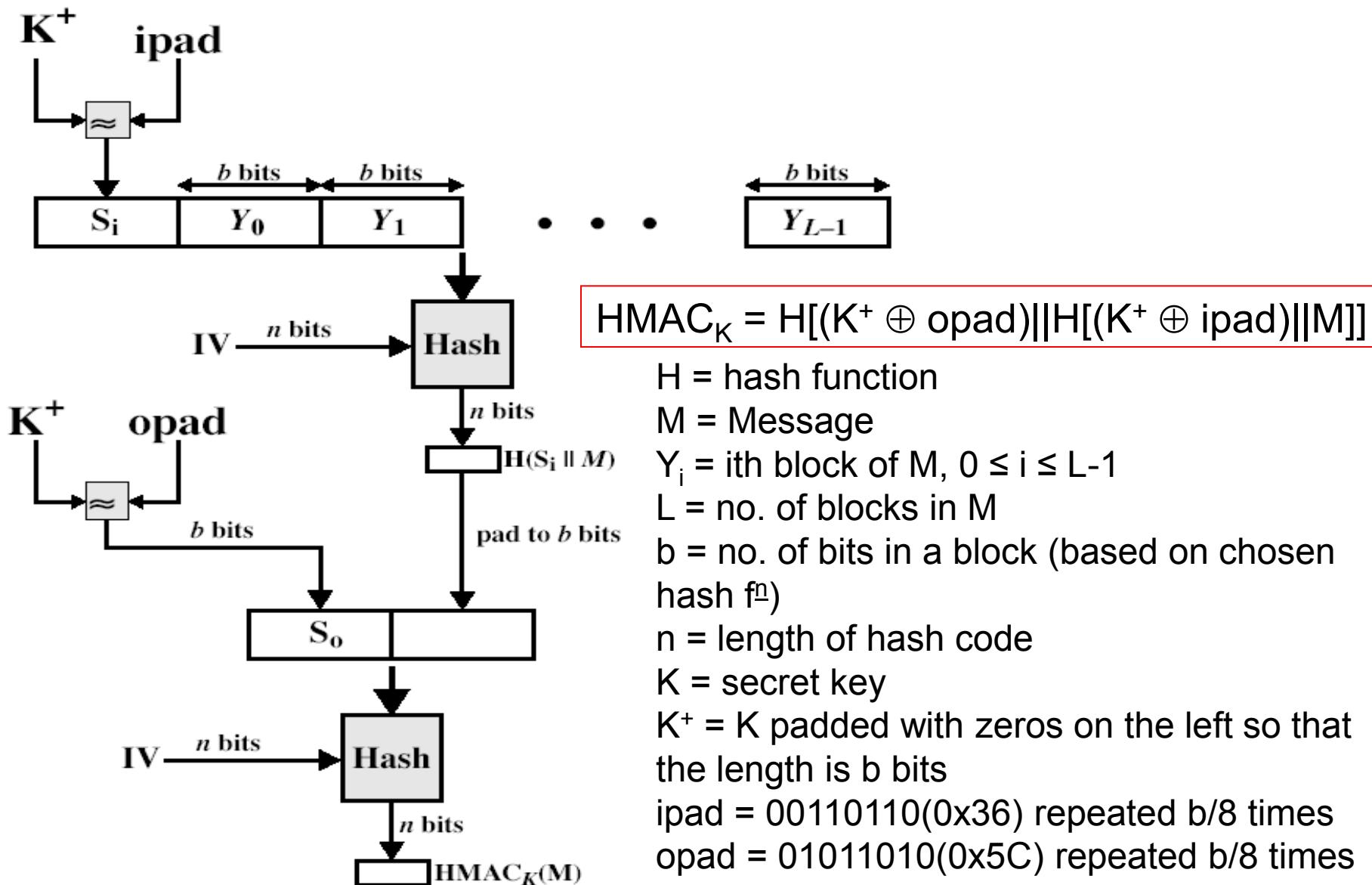
Other Secure HASH functions

	SHA-1	SHA-512	MD5	RIPEMD-160
Digest length	160 bits	512 bits	128 bits	160 bits
Basic unit of processing	512 bits	1024 bits	512 bits	512 bits
Number of steps	80 (4 rounds of 20)	80	64 (4 rnds of 16)	160 (5 paired rnds of 16)
Maximum message size	$2^{64}-1$ bits	$2^{128}-1$ bits	∞	∞

HMAC

- Uses a MAC derived from a cryptographic hash code, such as SHA-1.
- **Motivations:**
 - Cryptographic hash functions executes faster in software than encryption algorithms such as DES
 - Library code for cryptographic hash functions is widely available
 - No export restrictions from the US

HMAC Overview



HMAC Advantages

- Existing hash function can be implemented in HMAC
- Easy to replace with more secure or updated hash algorithm
- HMAC is proven more secure than hash algorithms

HMAC Security

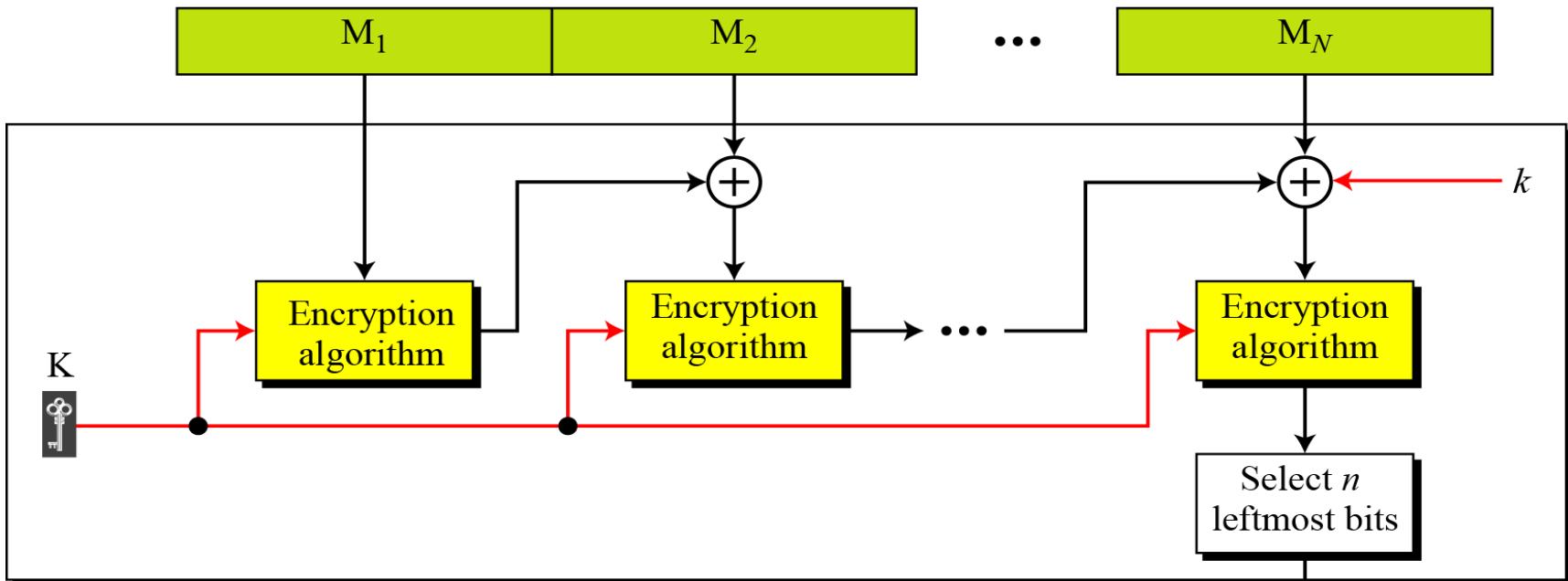
- Proved security of HMAC relates to that of the underlying hash algorithm
- Attacking HMAC requires either:
 - brute force attack on key used
 - birthday attack (but since keyed would need to observe a very large number of messages)
- Choose hash function used based on speed verses security constraints

CMAC (Cipher-based MAC)

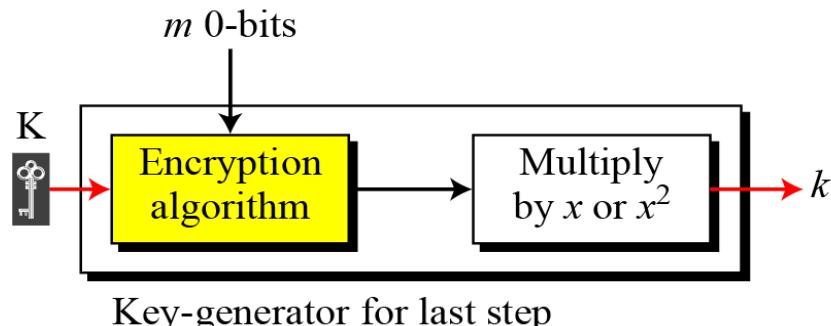
- “Hashless” MAC
 - Uses an encryption algorithm (DES, AES, etc.) to generate MAC
 - Based on same idea as cipher block chaining
- Compresses result to size of single block (unlike encryption)

CMAC Overview

M_i : Message block i



CMAC function



- Message broken into N blocks
- Each block fed into an encryption algorithm with key
- Result XOR'd with next block before encryption to make final MAC

CMAC Facts

- **Advantages:**
 - Can use existing encryption functions
 - Encryption functions have properties that resist pre-image and collision attacks
 - Cipher text designed to appear like “random noise”
 - good approximation of random oracle model
 - Most exhibit strong avalanche effect – minor change in message gives great change in resulting MAC
- **Disadvantage:**
 - Encryption algorithms (particularly when chained) can be much slower than hash algorithms

Cryptography and Network Security



Dr. B. Shanthini
Prof. & Head, IT Department
St. Peter's College of Engineering & Technology

Unit 3 – Outline

- Authentication requirement
- Authentication functions
- MAC
- Hash function
- Security of hash function and MAC
- MD5 and SHA
- HMAC and CMAC
- Digital signature and authentication protocols
- DSS – El Gamal – Schnorr

Authentication Protocols

- Used to convince parties of each others identity and to exchange session keys
- May be **one-way authentication** or **mutual authentication**
- Key issues are
 - **confidentiality** – to protect session keys
 - **timeliness** – to prevent replay attacks

Replay Attacks

- **Examples of replay attacks:**
 - simple replay (copies message and replays)
 - repetition that can be logged (within time frame)
 - repetition that cannot be detected (Actual msg suppressed)
 - backward replay without modification (to sender)
- **Countermeasures include**
 - use of sequence numbers (generally impractical)
 - timestamps (needs synchronized clocks)
 - challenge/response (using unique nonce)

Mutual Authentication Using Symmetric Encryption

- A two-level hierarchy of keys are used.
- Usually with a trusted Key Distribution Center (KDC)
 - each party shares own master key with KDC
 - KDC generates session keys used for connections between parties
 - master keys used to distribute the session keys to them

Needham-Schroeder Protocol

- Original third-party key distribution protocol
- For session between A and B mediated by KDC
- Protocol overview is:
 1. A→KDC : $ID_A \parallel ID_B \parallel N_1$
 2. KDC→A : $E_{Ka}[Ks \parallel ID_B \parallel N_1 \parallel E_{Kb}[Ks \parallel ID_A]]$
 3. A→B : $E_{Kb}[Ks \parallel ID_A]$
 4. B→A : $E_{Ks}[N_2]$
 5. A→B : $E_{Ks}[f(N_2)]$

Needham-Schroeder Protocol

- Used to securely distribute a new session key for communications between A & B
- But is vulnerable to a replay attack if an old session key has been compromised
 - then message 3 can be resent convincing B that is communicating with A
- Modifications to address this require:
 - timestamps
 - using an extra nonce

Mutual Authentication Using Public-Key Encryption

- Have a range of approaches based on the use of public-key encryption
- Need to ensure that they have correct public keys for other parties
- Using a central Authentication Server (AS)
- Various protocols exist using timestamps or nonces

Denning AS Protocol

- Denning presented the following:

1. A→AS : $ID_A \parallel ID_B$

2. AS→A : $E_{KRas}[ID_A \parallel KU_a \parallel T] \parallel$
 $E_{KRas}[ID_B \parallel KU_b \parallel T]$

3. A→B : $E_{KRas}[ID_A \parallel KU_a \parallel T] \parallel$
 $E_{KRas}[ID_B \parallel KU_b \parallel T] \parallel$
 $E_{KUb}[E_{KRa}[K_s \parallel T]]$

- Note session key is chosen by A, hence AS need not be trusted to protect it
- Timestamps prevent replay but require synchronized clocks

One-Way Authentication

- Required when sender & receiver are not in communications at same time (eg. email)
- Have header in clear so can be delivered by email system
- May want contents of body protected & sender authenticated

Using Symmetric Encryption

- Can refine use of KDC but can't have final exchange of nonces, vis:
 1. A→KDC : $ID_A \parallel ID_B \parallel N_1$
 2. KDC→A : $E_{Ka}[Ks \parallel ID_B \parallel N_1 \parallel E_{Kb}[Ks \parallel ID_A]]$
 3. A→B : $E_{Kb}[Ks \parallel ID_A] \parallel E_{Ks}[M]$
- Does not protect against replays
 - could rely on timestamp in message, though email delays make this problematic

Public-Key Approaches

- Have seen some public-key approaches
- If confidentiality is major concern, can use:
 $A \rightarrow B : E_{KUb}[Ks] \parallel E_{Ks}[M]$
 - has encrypted session key, encrypted message
- If authentication needed use a digital signature with a digital certificate:
 $A \rightarrow B : M \parallel E_{KRa}[H(M)] \parallel E_{KRas}[T \parallel ID_A \parallel KU_a]$
 - with message, signature, certificate

Digital Signatures

Inclusion: A conventional signature is included in the document; it is part of the document.

But when we sign a document digitally, we send the signature as a separate document.

Verification: For a CS, when the recipient receives a document, she compares the signature on the document with the signature on file.

For a DS, the recipient receives the message and the signature. The recipient needs to apply a verification technique to the combination of the message and the signature to verify the authenticity.

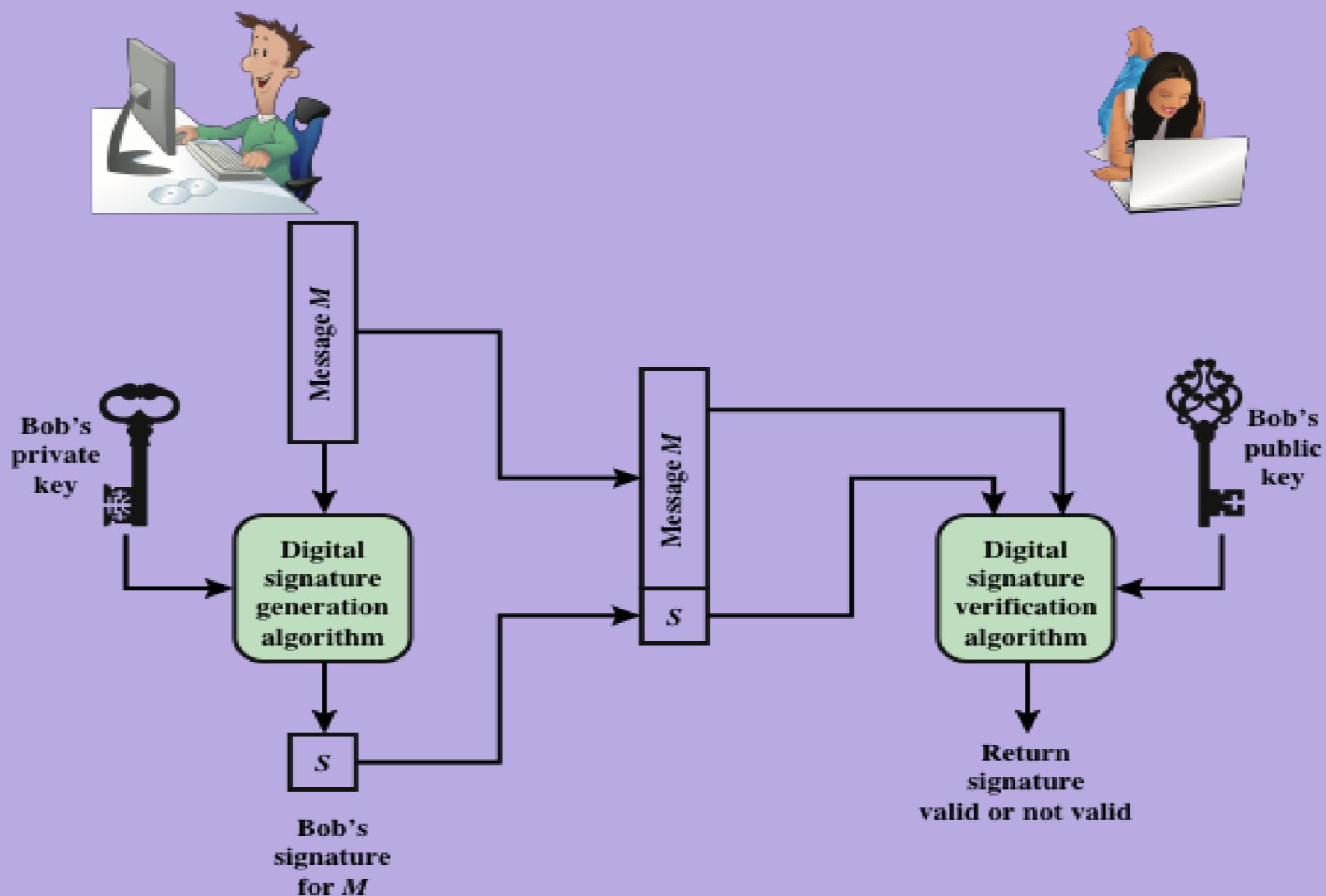
Digital Signatures

Relationship: For a CS, there is normally a one-to-many relationship between a signature and documents.

For a DS, there is a one-to-one relationship between a signature and a message.

Duplicity: In CS, a copy of the signed document can be distinguished from the original one on file.

In DS, there is no such distinction unless there is a factor of time on the document.

Bob**Transmit****Alice****Figure 13.1 Generic Model of Digital Signature Process**

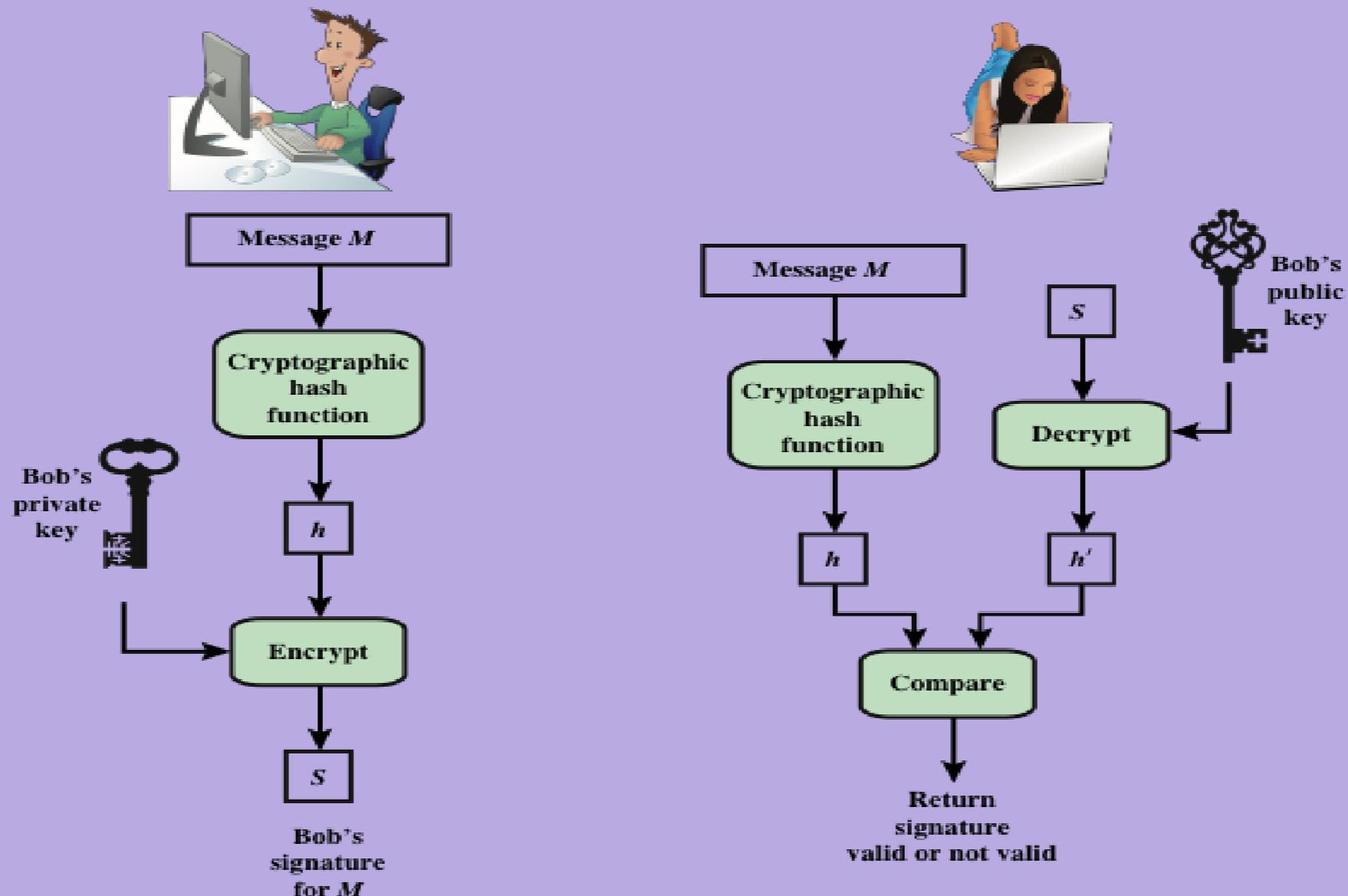
Bob**Alice**

Figure 13.2 Simplified Depiction of Essential Elements of Digital Signature Process

Digital Signature Properties

It must verify the author and the date and time of the signature

It must authenticate the contents at the time of the signature

It must be verifiable by third parties, to resolve disputes

Attacks

**Key-only
attack**

C only knows A's public key

**Known
message
attack**

C is given access to a set of messages and their signatures

**Generic
chosen
message
attack**

C chooses a list of messages before attempting to break A's signature scheme, independent of A's public key; C then obtains from A valid signatures for the chosen messages

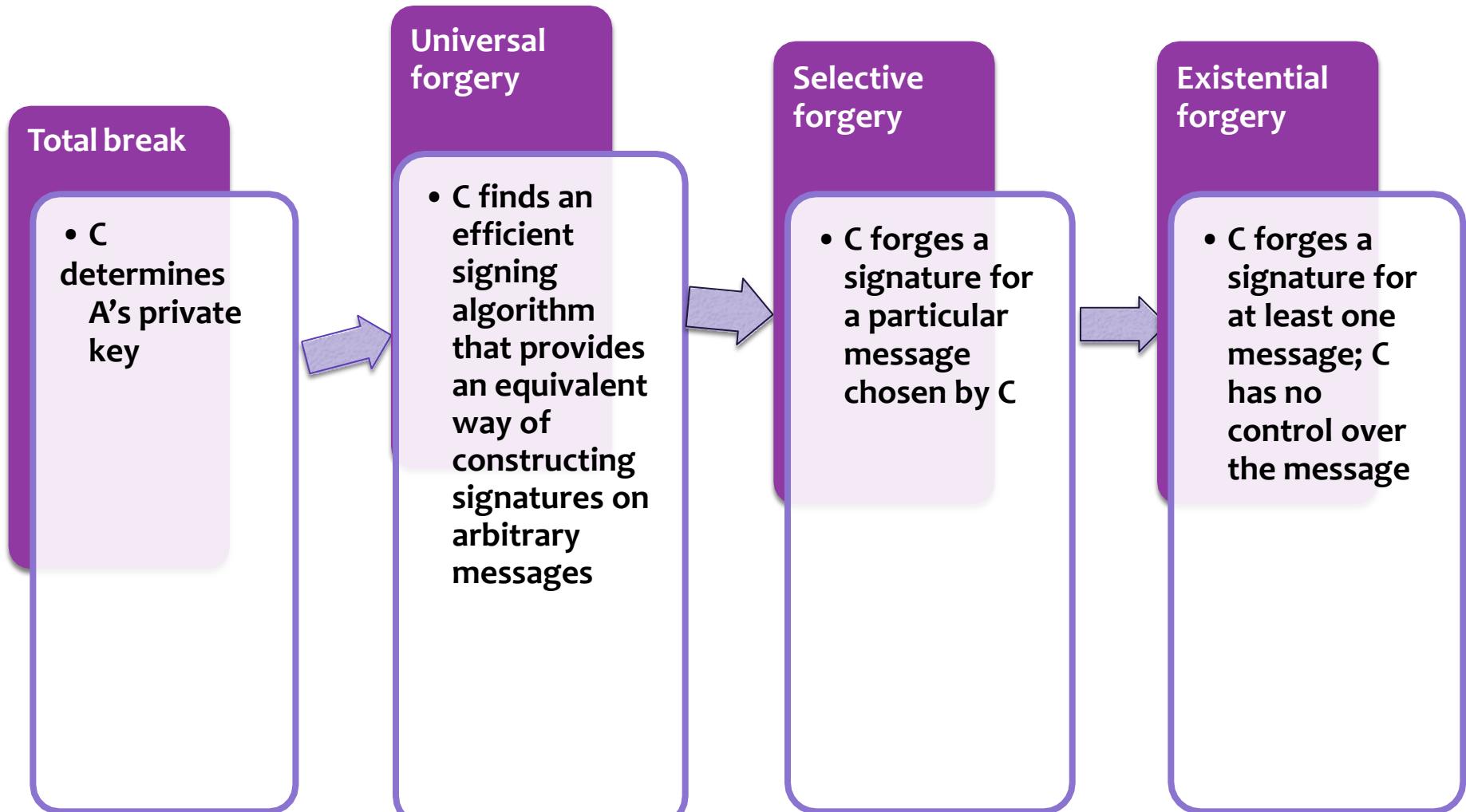
**Directed
chosen
message
attack**

Similar to the generic attack, except that the list of messages to be signed is chosen after C knows A's public key but before any signatures are seen

**Adaptive
chosen
message
attack**

C may request from A signatures of messages that depend on previously obtained message-signature pairs

Forgeries



Digital Signature Requirements

- The signature must be a bit pattern that depends on the message being signed
- The signature must use some information unique to the sender to prevent both forgery and denial
- It must be relatively easy to produce the digital signature
- It must be relatively easy to recognize and verify the digital signature
- It must be computationally infeasible to forge a digital signature, either by constructing a new message for an existing digital signature or by constructing a fraudulent digital signature for a given message
- It must be practical to retain a copy of the digital signature in storage

ElGamal Cryptosystem

- The ElGamal Algorithm provides an alternative to the RSA for public key encryption.
- Security of the RSA depends on the **difficulty of factoring large integers**.
- Security of the ElGamal depends on the **difficulty of computing discrete logs** in a large prime modulus.
- ElGamal has the **disadvantage** that the cipher text is twice as long as the plaintext.
- It has the **advantage** the same plaintext gives a different cipher text (with near certainty) each time it is encrypted.

ELGamal Cryptosystem

Alice chooses

- i) A large prime p_A (say 200 to 300 digits),
- ii) A primitive element α_A modulo p_A ,
- iii) A (possibly random) integer d_A with $2 \leq d_A \leq p_A - 2$.

Alice computes

$$\text{iv) } \beta_A \equiv \alpha_A^{d_A} \pmod{p_A}.$$

Alice's *public key* is (p_A, α_A, β_A) . Her *private key* is d_A .

ELGamal Cryptosystem

Bob encrypts a short message M ($M < p_A$) and sends it to Alice like this:

- i) Bob chooses a random integer k (which he keeps secret).
- ii) Bob computes $r \equiv \alpha_A^k \pmod{p_A}$ and $t \equiv \beta_A^k M \pmod{p_A}$, and then discards k .

Bob sends his encrypted message (r, t) to Alice.

ELGamal Cryptosystem

When Alice receives the encrypted message (r, t) , she decrypts (using her private key d_A) by computing tr^{-d_A} .

Note

$$\begin{aligned}
 tr^{-d_A} &\equiv \beta_A^k M (\alpha_A^k)^{-d_A} \pmod{p_A} \\
 &\equiv (\alpha_A^{d_A})^k M (\alpha_A^k)^{-d_A} \pmod{p_A} \\
 &\equiv M \pmod{p_A}
 \end{aligned}$$

ELGamal Cryptosystem

Even if Eve intercepts the ciphertext (r, t) , she cannot perform the calculation above because she doesn't know d_A .

$$\beta_A \equiv \alpha_A^{d_A} \pmod{p_A}, \text{ so } d_A \equiv L_{\alpha_A}(\beta_A)$$

Eve can find d_A if she can compute a discrete log in the large prime modulus p_A , presumably a computation that is too difficult to be practical.

Caution: Bob should choose a different random integer k for each message he sends to Alice.

ElGamal Example

- $p_A = 11$ and $\alpha A = 2$
- Alice computes her key:
 - chooses $d_A = 5$ & computes $\beta_A = 2^5 \text{ mod } 11 = 10$
 - Public Key = $(11, 2, 10)$
 - Private Key = (5)
- Bob send message $m=1$ as $(9, 1)$ by
 - choosing random $k=6$
 - computing $r = \alpha_A^k \text{ mod } p_A = 2^6 \text{ mod } 11 = 9$
 - computing $t = \beta_A^k \cdot m \text{ mod } p_A = 10^6 \cdot 1 \text{ mod } 11 = 1$
- Alice recovers original message by computing:
 - $m = tr^{-d_A} \text{ mod } p_A = 1 \cdot 9^{10-5} \text{ mod } 11 = 1$

ElGamal Example

- $p_A = 19$ and $\alpha A = 10$
- Alice computes her key:
 - A chooses $d_A = 5$ & computes $\beta_A = 10^5 \bmod 19 = 3$
 - Public Key = $(19, 10, 3)$
 - Private Key = (5)
- Bob send message $m = 17$ as $(11, 5)$ by
 - choosing random $k = 6$
 - computing $r = \alpha_A^k \bmod p_A = 10^6 \bmod 19 = 11$
 - computing $t = \beta_A^k \cdot m \bmod p_A = 3^6 \cdot 17 \bmod 19 = 5$
- Alice recovers original message by computing:
 - $m = t r^{-d_A} \bmod p_A = 5 \cdot 11^{18-5} \bmod 19 = 17$

Schnorr Digital Signature

- Scheme is based on discrete logarithms
- Minimizes the message-dependent amount of computation required to generate a signature
- Multiplying a $2n$ -bit integer with an n -bit integer
- Main work can be done during the idle time of the processor
- Based on using a prime modulus p , with $p - 1$ having a prime factor q of appropriate size
- Typically p is a 1024-bit number, and q is a 160-bit number

Schnorr Digital Signature

Generation of Private-Public Key Pair:

1. Choose prime p and q , such that q is a prime factor of $p-1$.
2. Choose an integer a , such that $a^q \equiv 1 \pmod{p}$.
(a , p and q are the global public keys and common to all users of a group)
3. Choose a random integer s with $0 < s < q$.
(This ‘ s ’ is user’s private key)
4. Calculate $v = a^{-s} \pmod{p}$.
(This ‘ v ’ is user’s public key)

Schnorr Digital Signature

Generation of signature with Private-Public Key Pair (s-v):

1. Choose a random integer r with $0 < r < q$ and compute $x = a^r \bmod p$.
2. Concatenate the message M with x and hash the result to compute the value $e = H(M||x)$
3. Compute $y = (r+se) \bmod q$.

The signature consists of the pair (e, y)

Schnorr Digital Signature

Signature verification by any other user:

1. Compute $x' = a^y v^e \pmod{p}$.
2. Verify that $e = H(M||x')$

To Prove this:

$$x' = a^y v^e = a^y a^{-se} = a^{y-se} = a^r = x \pmod{p}$$

$$\text{Hence } H(M||x') = H(M||x)$$

Cryptography and Network Security



Dr. B. Shanthini
Prof. & Head

St. Peter's College of Engineering & Technology 1

Unit 4 – Outline

- Authentication applications:
 - Kerberos
 - X.509
- Authentication services
- Internet Firewalls for Trusted System:
 - Roles of Firewalls
 - Firewall related terminology
 - Types of Firewalls
 - Firewall designs
 - Firewalls design principles.
- SET for E-Commerce Transactions
- Intruder
 - Intrusion detection system
- Virus and related threats
 - Countermeasures
- Trusted systems
- Practical implementation of cryptography and security.

Security Concerns

- Key concerns are **confidentiality** and **timeliness**
- To provide confidentiality must encrypt identification and session key info
- Which requires the use of previously shared private or public keys
- Need timeliness to prevent **replay attacks**
- Provided by using sequence numbers or timestamps or challenge/response

KERBEROS



In Greek mythology, a many headed dog, the guardian of the entrance of Hades

KERBEROS

- Users wish to access services on servers.
- Three threats exist:
 - User pretends to be another user.
 - User alter the network address of a workstation.
 - User eavesdrop on exchanges and use a replay attack.

Kerberos

- Provides a centralized authentication server to authenticate users to servers and servers to users.
- Relies on conventional encryption, making no use of public-key encryption
- Two versions: version 4 and 5
- Version 4 makes use of DES

Kerberos

- Motivation: 3 Approaches
 1. Rely on each individual client to assure the identity of the user to enforce security policy based on user identification.
 2. Require that client systems authenticate themselves to servers
 3. Require that the user to prove his or her identity for each service invoked

Kerberos

- Motivation: Requirements

- **Secure:** Eavesdropper should not be able to obtain the necessary info. to impersonate a user.
- **Reliable:** Should be highly reliable and should employ a distributed server architecture with one system able to back up another.
- **Transparent:** User should not be aware that authentication is taking place beyond the requirement to enter the password.
- **Scalable:** System should be capable of supporting large numbers of clients and servers.

Kerberos Version 4

- Terms:
 - C = Client
 - AS = authentication server
 - V = server
 - ID_c = identifier of user on C
 - ID_v = identifier of V
 - P_c = password of user on C
 - AD_c = network address of C
 - Kv = secret encryption key shared by AS and V
 - TS = timestamp
 - || = concatenation

A Simple Authentication Dialogue

- | | |
|-------------|--|
| (1) C → AS: | ID _c P _c ID _v |
| (2) AS → C: | Ticket |
| (3) C → V: | ID _c Ticket |

Ticket = E_{K_v}[ID_c || P_c || ID_v]

Version 4 Authentication Dialogue

- **Problems:**
 - User would need a new ticket for every different service.
 - Since the plaintext transmission of the password is involved, an eavesdropper could capture the pw and use any accessible service.

Lifetime associated with the ticket-granting ticket

If too short → repeatedly asked for password

If too long → greater opportunity to replay

Ticket Granting Server Scenario

Once Per User Logon Session

1. C → AS: $ID_C \parallel ID_{tgs}$

2. AS → C: $E(K_C, Ticket_{tgs})$

Once Per Type of Service

3. C → TGS: $ID_C \parallel ID_V \parallel Ticket_{tgs}$

4. TGS → C: $Ticket_V$

Once Per Service Session

5. C → V: $ID_C \parallel Ticket_V$

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

$Ticket_V = E(K_V, [ID_C \parallel AD_c \parallel ID_V \parallel TS_2 \parallel Lifetime_2])$

Ticket Granting Server Scenario

- Problems:
 - Lifetime associated with the ticket-granting ticket
 - If too short → repeatedly asked for password
 - If too long → greater opportunity to replay
 - There may be a requirement for servers to authenticate themselves to users.

Version 4 Authentication Dialogue

Authentication Service Exchange: To obtain Ticket-Granting Ticket

- (1) $C \rightarrow AS: ID_c \parallel ID_{tgs} \parallel TS_1$
- (2) $AS \rightarrow C: E_{K_c}[K_{c,tgs} \parallel ID_{tgs} \parallel TS_2 \parallel \text{Lifetime}_2 \parallel \text{Ticket}_{tgs}]$

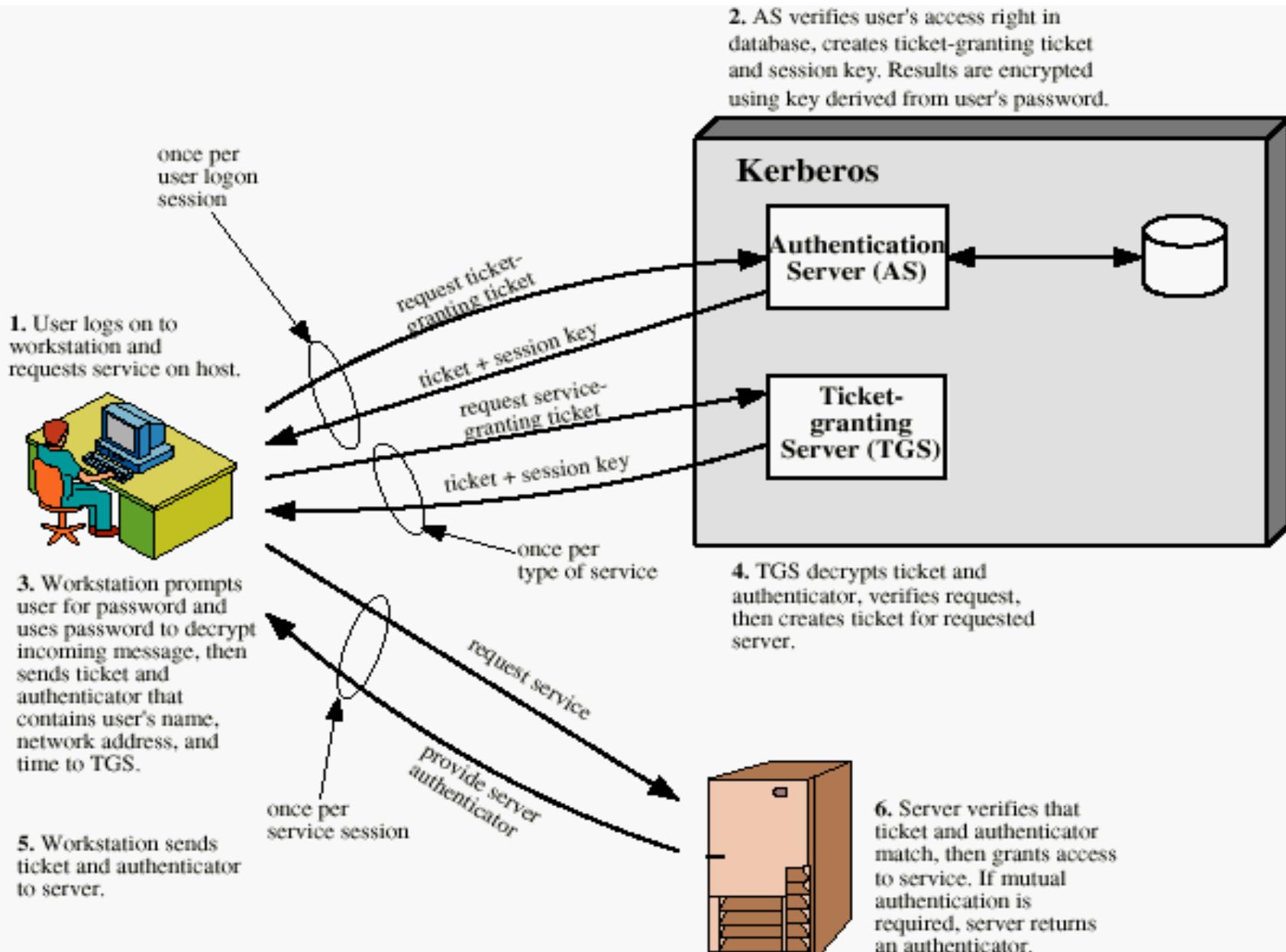
Ticket-Granting Service Exchange: To obtain Service-Granting Ticket

- (3) $C \rightarrow TGS: ID_v \parallel \text{Ticket}_{tgs} \parallel \text{Authenticator}_c$
- (4) $TGS \rightarrow C: E_{K_c}[K_{c,v} \parallel ID_v \parallel TS_4 \parallel \text{Ticket}_v]$

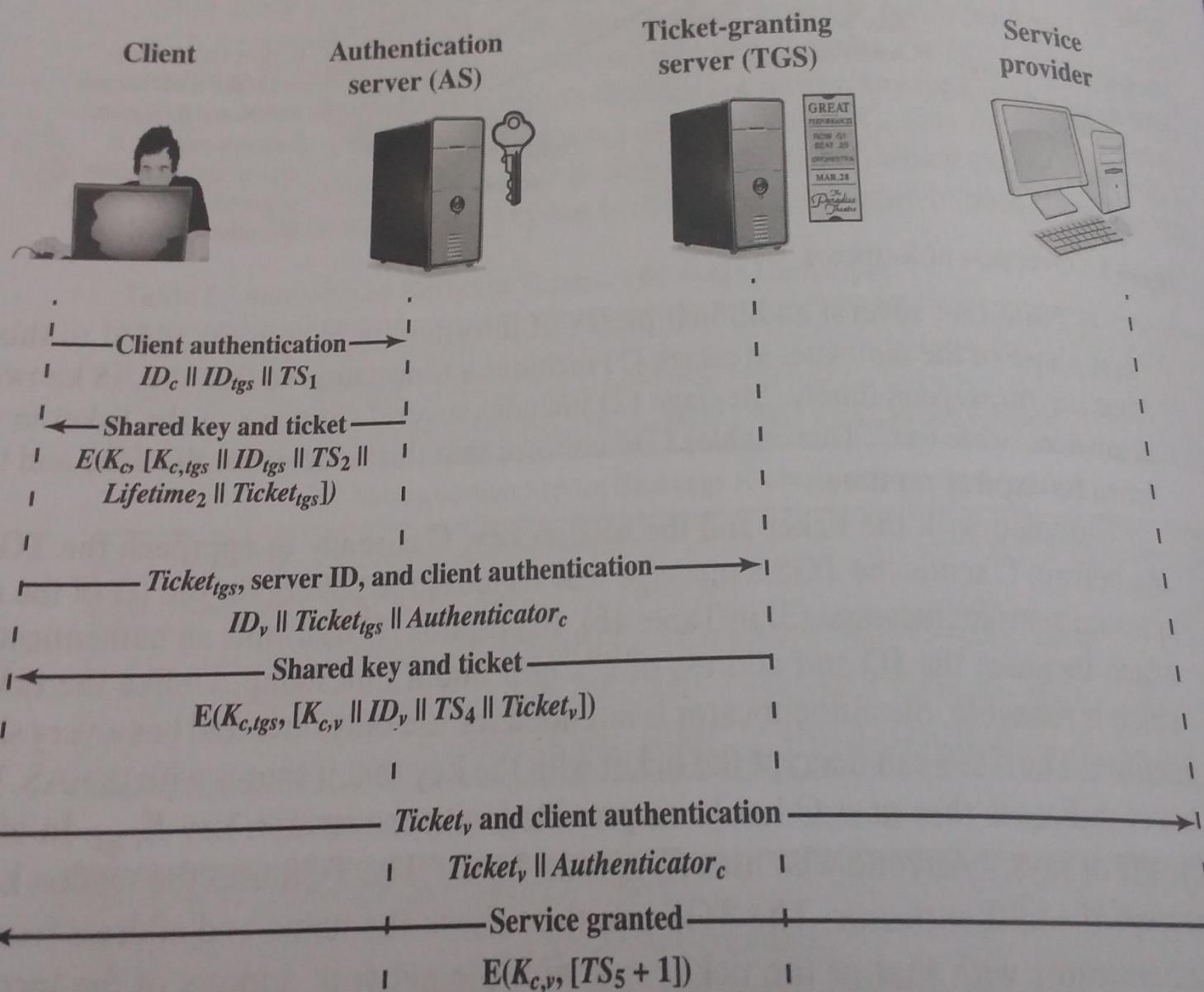
Client/Server Authentication Exchange: To Obtain Service

- (5) $C \rightarrow V: \text{Ticket}_v \parallel \text{Authenticator}_c$
- (6) $V \rightarrow C: E_{K_c,v}[TS_5 + 1]$

Overview of Kerberos



Kerberos Exchanges



Kerberos Realms & Multiple Kerberi

A Full-Service Kerberos Environment called **Kerberos Realm** consists:

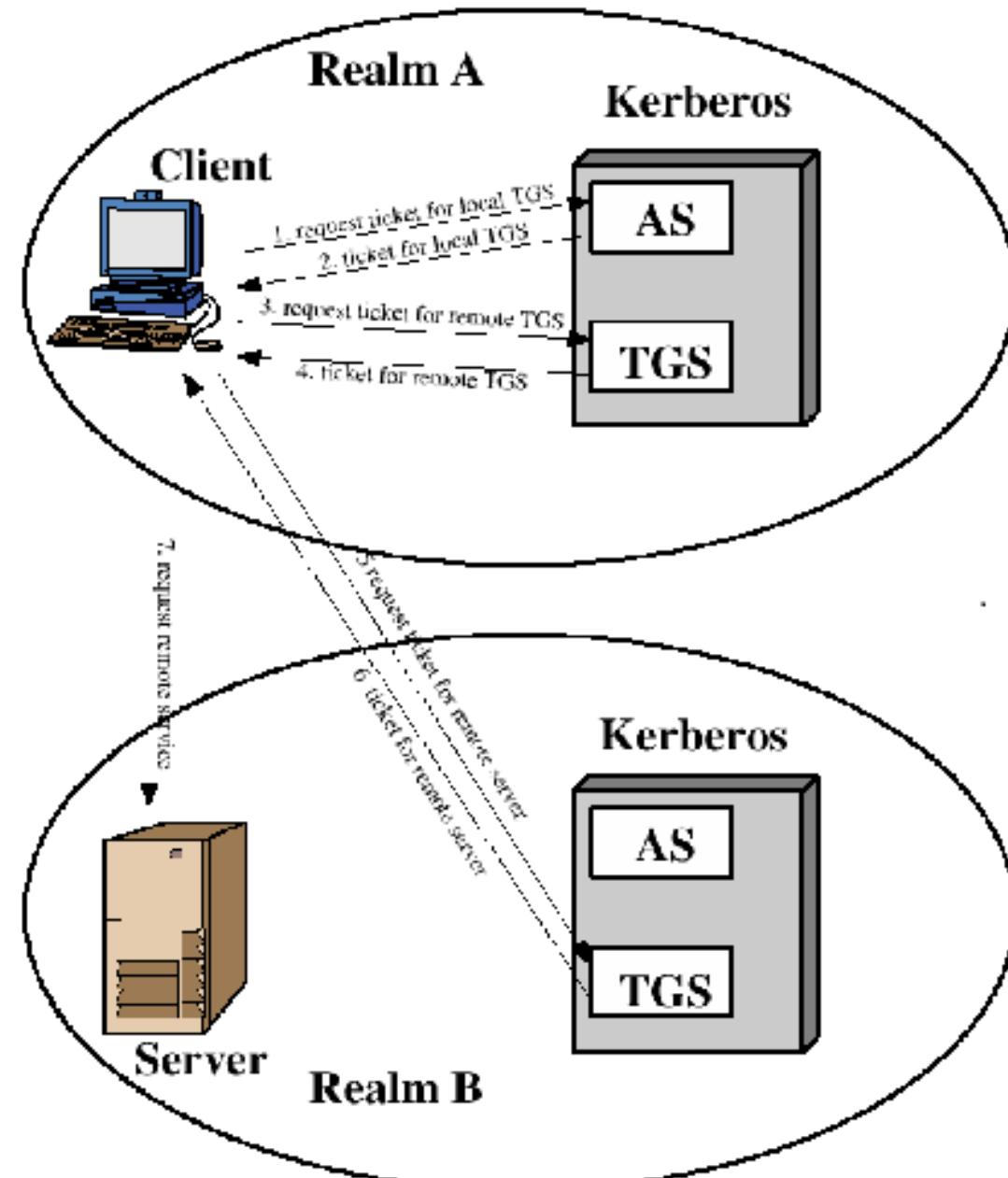
1. Kerberos Server
2. No. of Clients
3. No. of Application Servers

Kerberos Realms & Multiple Kerberi

Application Servers Requirements:

1. Kerberos Server must have the user ID and hashed passwords of all participating users in its database. All users are registered with the Kerberos Server.
2. Kerberos Server must share a secret key with each server. All servers are registered with the Kerberos server.
3. The Kerberos Server in each interoperating realm shares a secret key with the server in other realm.
2 Kerberos servers are registered with each other.

Request for Service in Another Realm



1. Request ticket for local TGS
2. Ticket for local TGS
3. Request ticket for remote TGS
4. Ticket for remote TGS
5. Request ticket for remote server
6. Ticket for remote server
7. Request for remote service

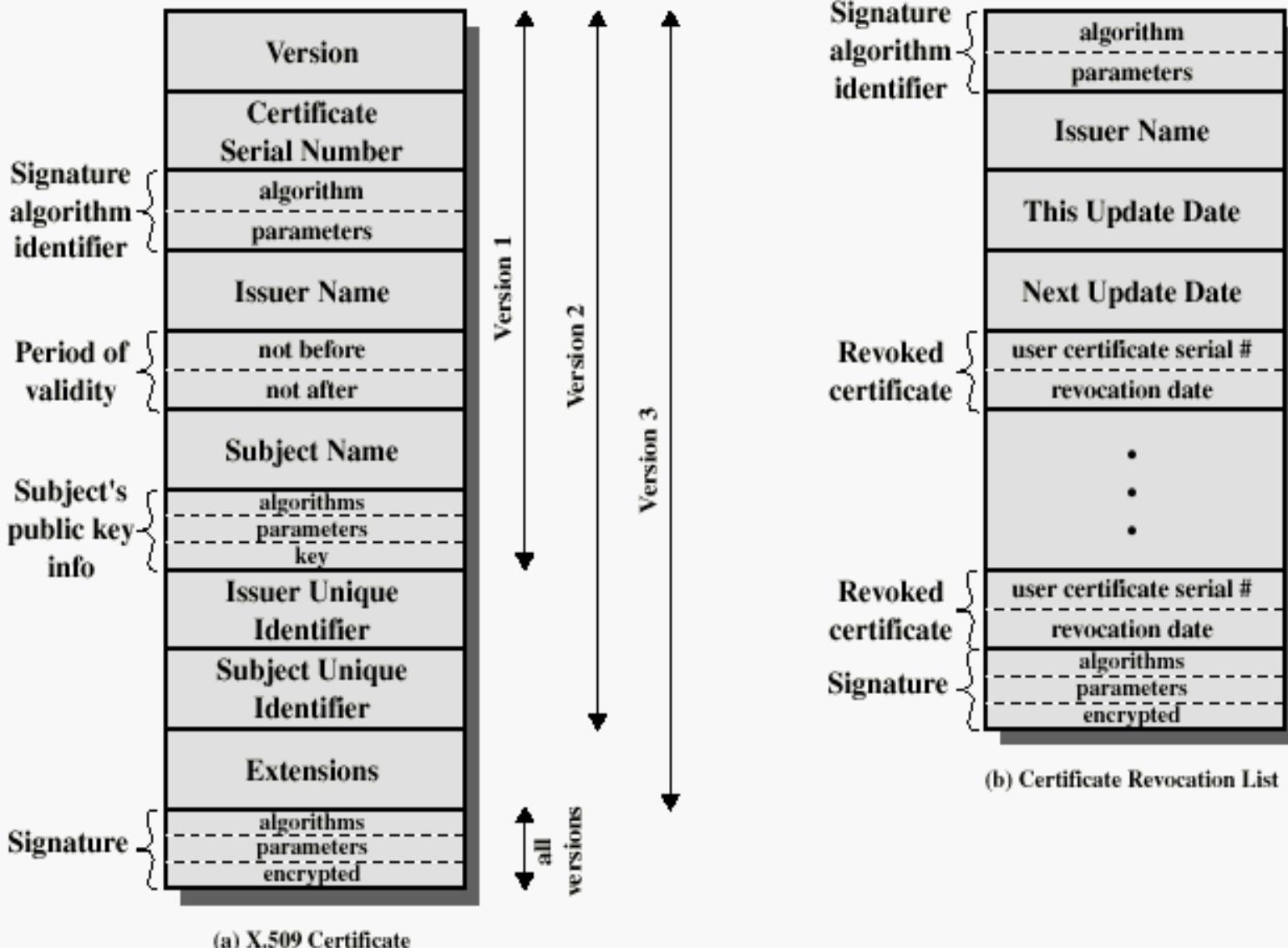
Difference Between Version 4 and 5

- Encryption system dependence (V.4 DES)
- Internet protocol dependence
- Message byte ordering
- Ticket lifetime
- Authentication forwarding
- Interrealm authentication

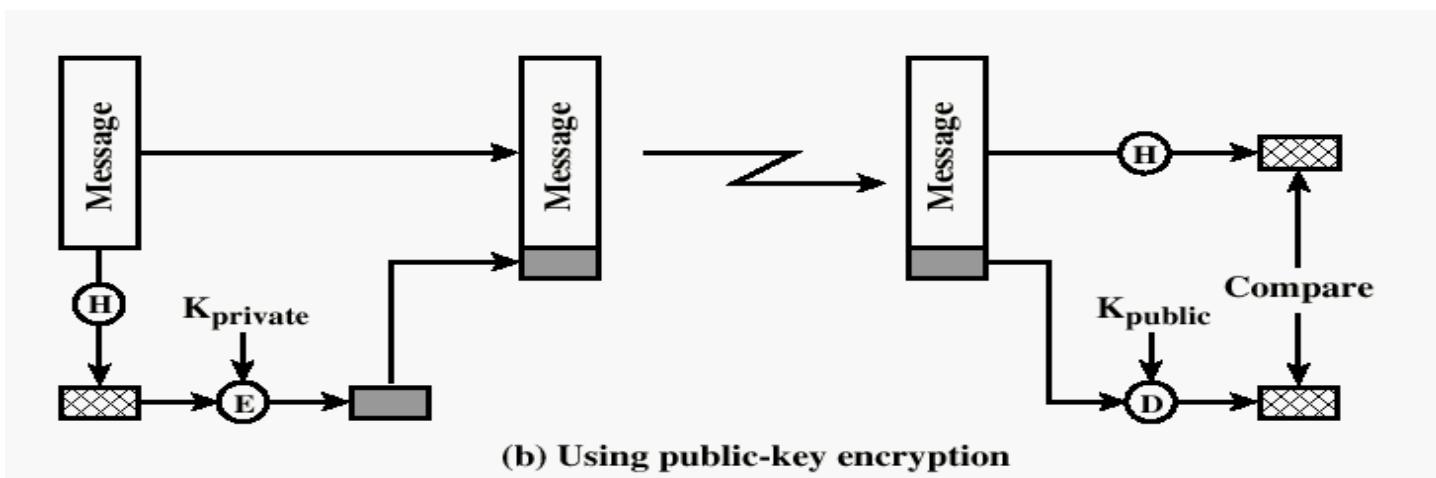
X.509 Authentication Service

- Distributed set of servers that maintains a database about users.
- Each certificate contains the public key of a user and is signed with the private key of a CA.
- Is used in S/MIME, IP Security, SSL/TLS and SET.
- RSA is recommended to use.

X.509 Formats



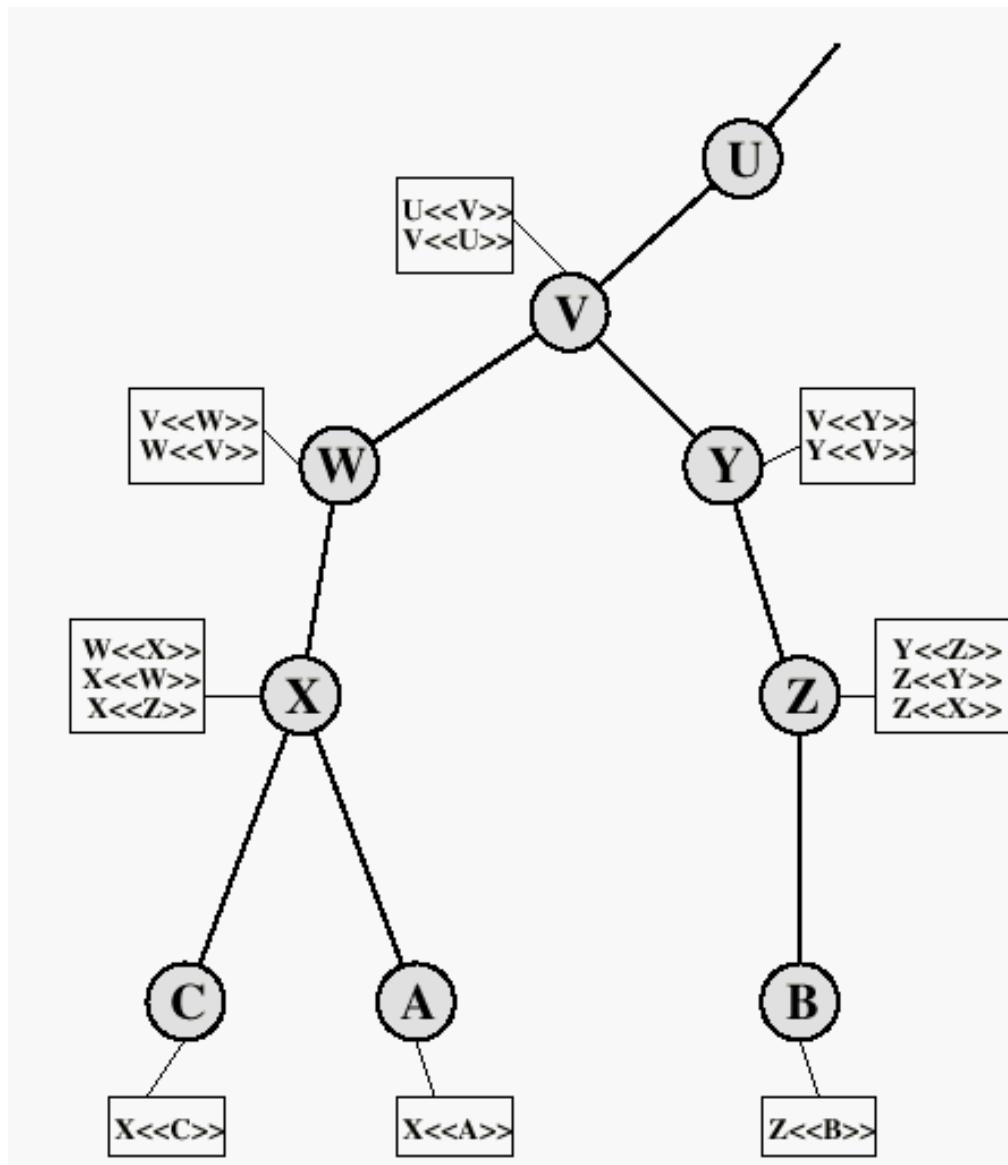
Typical Digital Signature Approach



Obtaining a User's Certificate

- Characteristics of certificates generated by CA:
 - Any user with access to the public key of the CA can recover the user public key that was certified.
 - No part other than the CA can modify the certificate without this being detected.

X.509 CA Hierarchy



Revocation of Certificates

- Reasons for revocation:
 - The user's secret key is assumed to be compromised.
 - The user is no longer certified by this CA.
 - The CA's certificate is assumed to be compromised.

Authentication Procedures

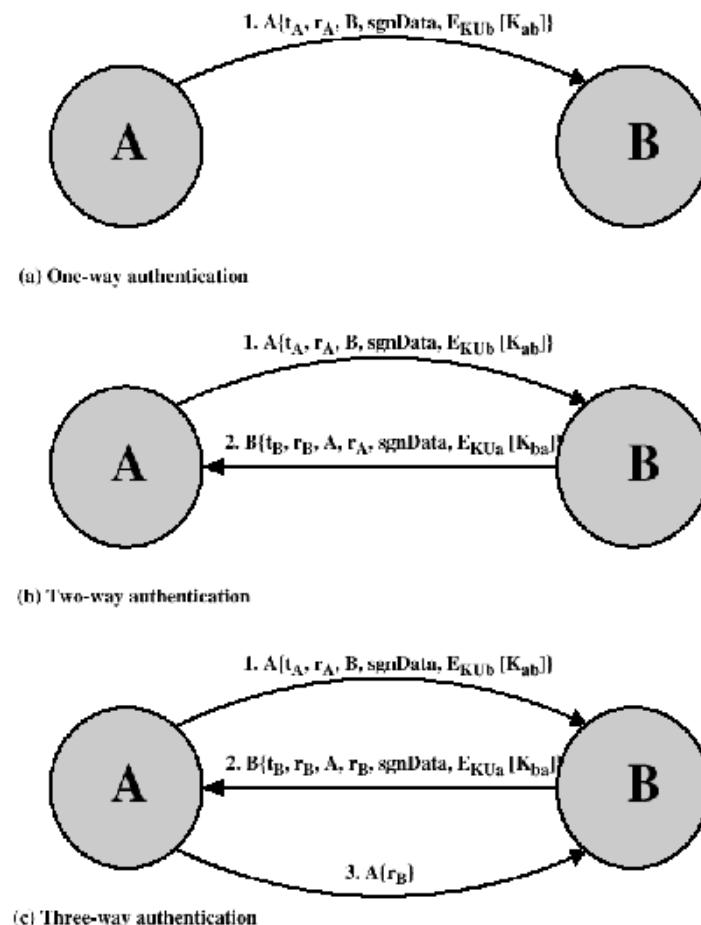


Figure 4.5 X.509 Strong Authentication Procedures

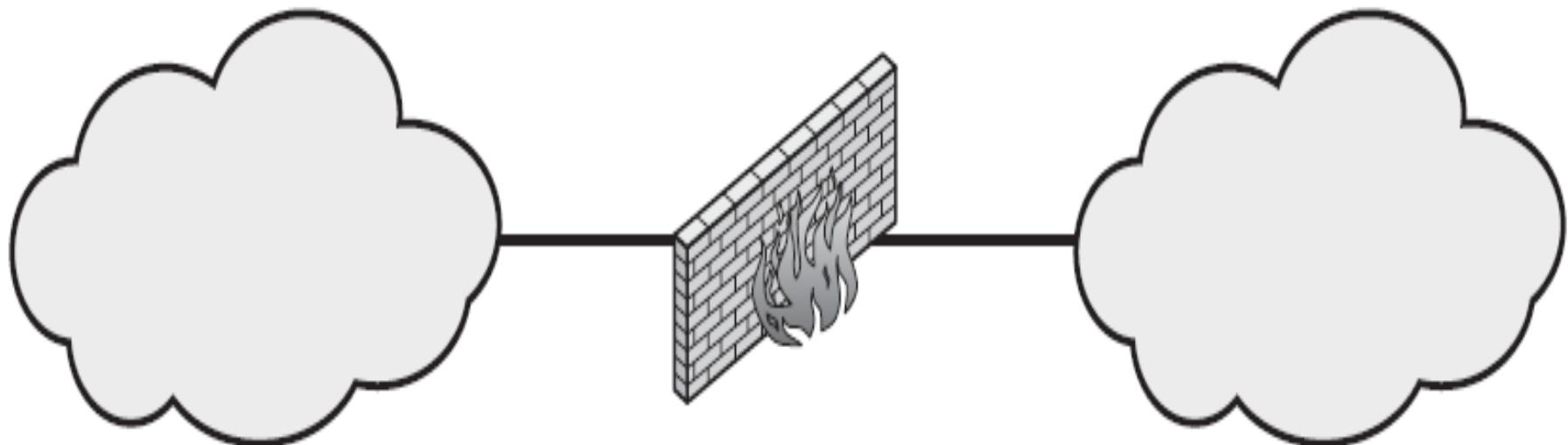
Firewalls

Firewall

Internal (protected) network
(e.g., enterprise network)

External (untrusted) network
(e.g., Internet)

Firewall



Need for Firewalls

- **Centralized data processing system**, with a central mainframe supporting a no. of directly connected terminals
- **Local area networks (LANs)** interconnecting PCs and terminals to each other and the mainframe
- **Premises network**, consisting of a number of LANs, interconnecting PCs, servers, and a mainframe or two
- **Enterprise-wide network**, consisting of multiple, geographically distributed premises networks interconnected by a private wide area network (WAN)
- **Internet connectivity**, in which the various premises networks all hook into the Internet and may or may not also be connected by a private WAN

Design Goals of Firewalls

- 1. All traffic from inside to outside, and vice versa, must pass through the firewall.**
 - This is achieved by physically blocking all access to the local network except via the firewall.
- 2. Only authorized traffic, as defined by the local security policy, will be allowed to pass.**
 - Various types of firewalls are used, which implement various types of security policies.
- 3. The firewall itself is immune to penetration.**
 - This implies the use of a hardened system with a secured operating system.

Characteristics of Firewalls

- **Service control:** Determines the types of Internet services that can be accessed, inbound or outbound. The firewall may filter traffic on the basis of IP address, protocol, or port number.
- **Direction control:** Determines the direction in which particular service requests may be initiated and allowed to flow through the firewall.
- **User control:** Controls access to a service according to which user is attempting to access it.
- **Behavior control:** Controls how particular services are used. For example, the firewall may filter e-mail to eliminate spam, or it may enable external access to only a portion of the information on a local Web server.

Capabilities of Firewalls

1. A **firewall defines a single choke point** that keeps unauthorized users out of the protected network, prohibits potentially vulnerable services from entering or leaving the network, and provides protection from various kinds of IP spoofing and routing attacks.
2. A **firewall provides a location for monitoring security-related events** like auditing and alarms.
3. A **firewall is a convenient platform for several Internet functions that are not security related** like network address translator and network mgmt. function.
4. A **firewall can serve as the platform for IPSec and virtual private networks.**

Limitations of Firewalls

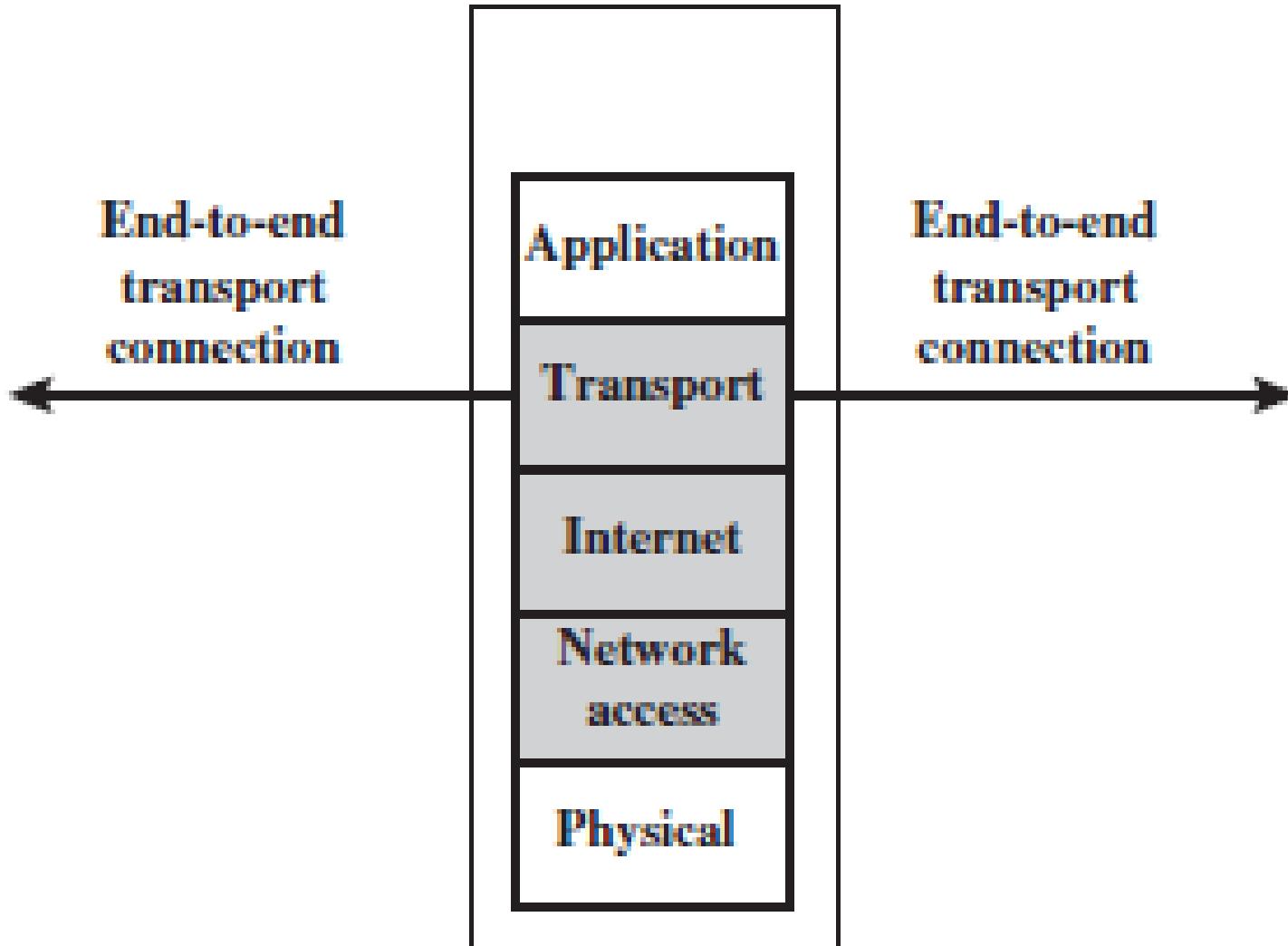
- 1. The firewall cannot protect against attacks that bypass the firewall.**
- 2. The firewall may not protect fully against internal threats, such as a disgruntled employee or an employee who unwittingly cooperates with an external attacker.**
- 3. An improperly secured wireless LAN may be accessed from outside the organization.**
- 4. A laptop, PDA, or portable storage device may be used and infected outside the corporate network, and then attached and used internally.**

Types of Firewalls

1. Packet Filtering Firewall

- A firewall may act as a **packet filter**.
- It can operate as a **positive filter**, allowing to pass only packets that meet specific criteria, or as a **negative filter**, rejecting any packet that meets certain criteria.
- A packet filtering firewall applies a **set of rules** to each incoming and outgoing IP packet and then **forwards or discards** the packet.
- The firewall is typically configured to filter packets going in **both directions**.

1. Packet Filtering Firewall



1. Packet Filtering Firewall

- Filtering rules are based on info. in network packet:
 - **Source IP address:** The IP address of the system that originated the IP packet (e.g., 192.178.1.1)
 - **Destination IP address:** The IP address of the system the IP packet is trying to reach (e.g., 192.168.1.2)
 - **Source and destination transport-level address:** The transport-level (e.g., TCP or UDP) port number
 - **IP protocol field:** Defines the transport protocol
 - **Interface:** For a firewall with three or more ports, which interface of the firewall the packet came from or which interface of the firewall the packet is destined for

1. Packet Filtering Firewall

- The packet filter is typically set up as a **list of rules** based on matches to fields in the IP or TCP header.
- If there is a **match** to one of the rules, that rule is invoked to **forward or discard** the packet.
- If there is **no match** to any rule, then a **default action** is taken.
- Two default policies are possible:
 - **Default = discard:** That which is not expressly permitted is prohibited.
 - **Default = forward:** That which is not expressly prohibited is permitted.

1. Packet Filtering Firewall

- Following table gives some examples of packet filtering rule sets.
- In each set, the rules are applied top to bottom.
- The “*” in a field is a wildcard designator that matches everything.
- We assume that the default = discard policy is in force.

action	ourhost	port	theirhost	port	comment
block	*	*	SPIGOT	*	we don't trust these people
allow	OUR-GW	25	*	*	connection to our SMTP port

Rule Set B

action	ourhost	port	theirhost	port	comment
block	*	*	*	*	default

Rule Set C

action	ourhost	port	theirhost	port	comment
allow	*	*	*	25	connection to their SMTP port

Rule Set D

action	src	port	dest	port	flags	comment
allow	{our hosts}	*	*	25		our packets to their SMTP port
allow	*	25	*	*	ACK	their replies

Rule Set E

action	src	port	dest	port	flags	comment
allow	{our hosts}	*	*	*		our outgoing calls
allow	*	*	*	*	ACK	replies to our calls
allow	*	*	*	>1024		traffic to nonservers

Packet Filtering Firewall-Rule Set

- A. Inbound mail is allowed, but only to a gateway host, however, packets from SPIGOT, are blocked because that host has a history of sending massive files in e-mail messages.
- B. This is an explicit statement of the default policy.
- C. This rule set is intended to specify that any inside host can send mail to the outside.
- D. This rule set achieves the intended result that was not achieved in C.
- E. This rule set is one approach to handling FTP connections.

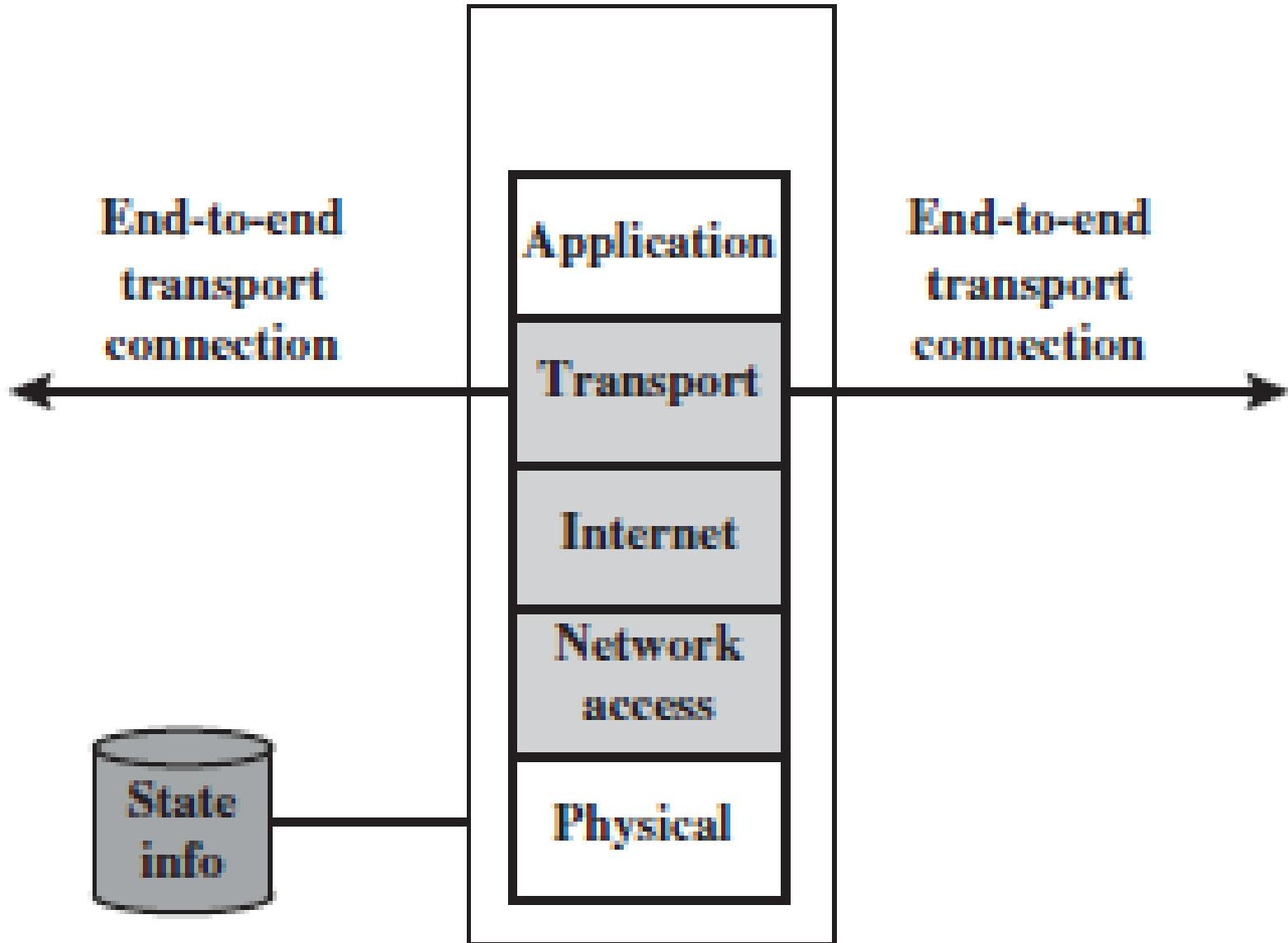
Attacks on Packet Filtering Firewall

- **IP address spoofing:** The intruder transmits packets from the outside with a source IP address field containing an address of an internal host.
 - The **countermeasure** is to discard packets with an inside source address if the packet arrives on an external interface.
- **Source routing attacks:** The source station specifies the route that a packet should take as it crosses the Internet, that this will bypass security measures that do not analyze the source routing information.
 - The **countermeasure** is to discard all packets that use this option.

Attacks on Packet Filtering Firewall

- **Tiny fragment attacks:** The intruder uses the IP fragmentation option to create extremely small fragments and force the TCP header information into a separate packet fragment.
 - **Countermeasure:** A tiny fragment attack can be defeated by enforcing a rule that the first fragment of a packet must contain a predefined minimum amount of the transport header. If the first fragment is rejected, the filter can remember the packet and discard all subsequent fragments.

2. Stateful Inspection Firewall



2. Stateful Inspection Firewall

- A **packet filtering firewall** permit inbound network traffic on all high-numbered ports for TCP-based traffic to occur.
- This creates a **vulnerability** that can be exploited by unauthorized users.
- A **stateful inspection packet firewall** tightens up the rules for TCP traffic by creating a **directory** of currently established outbound TCP connections
 - The packet filter will now allow incoming traffic to high-numbered ports only for those packets that fit the profile of one of the entries in this directory.

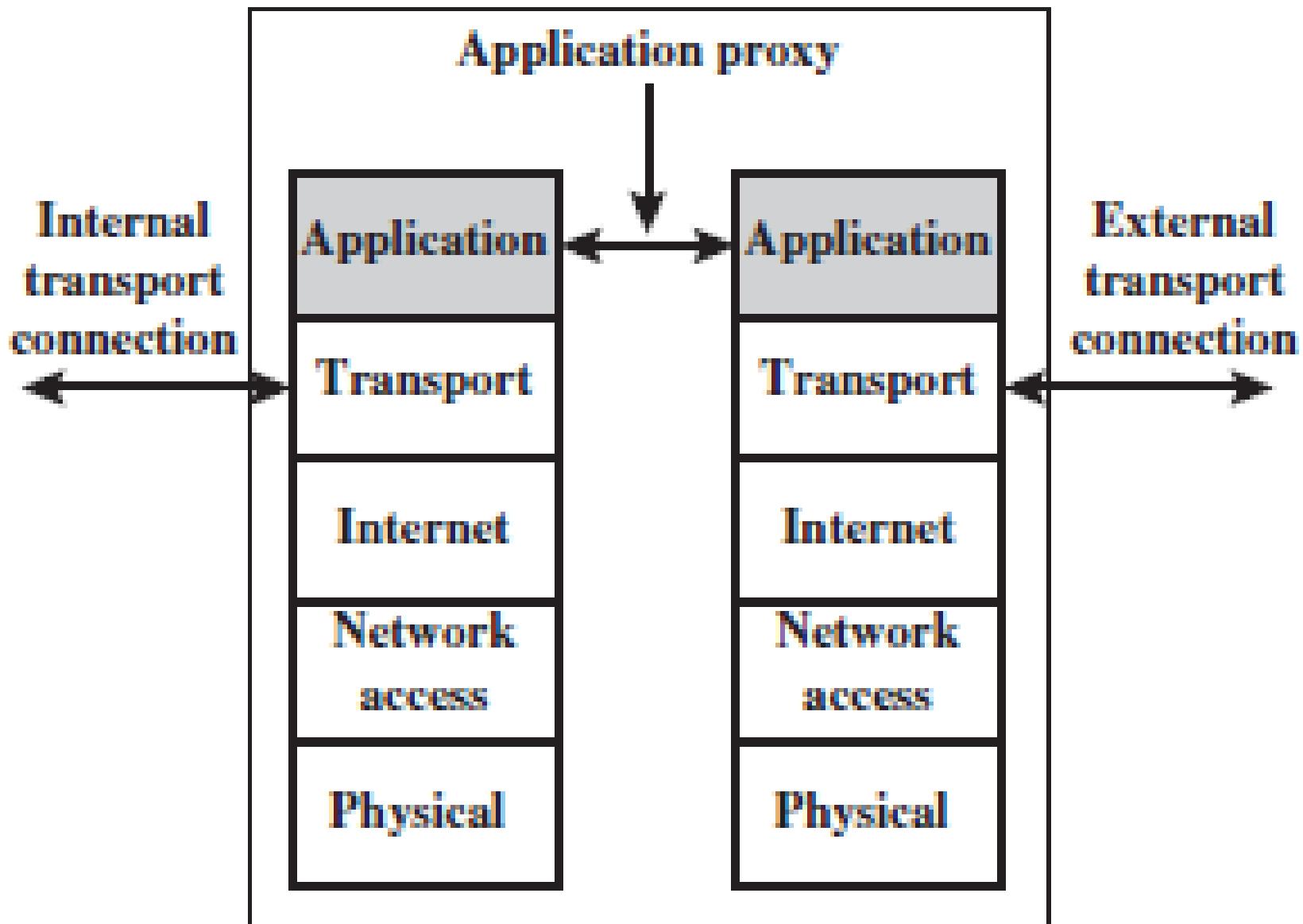
2. Stateful Inspection Firewall

Source Address	Source Port	Destination Address	Destination Port	Connection State
192.168.1.100	1030	210.22.88.29	80	Established
192.168.1.102	1031	216.32.42.123	80	Established
192.168.1.101	1033	173.66.32.122	25	Established
192.168.1.106	1035	177.231.32.12	79	Established
223.43.21.231	1990	192.168.1.6	80	Established
2122.22.123.32	2112	192.168.1.6	80	Established
210.922.212.18	3321	192.168.1.6	80	Established
24.102.32.23	1025	192.168.1.6	80	Established
223.21.22.12	1046	192.168.1.6	80	Established

2. Stateful Inspection Firewall

- A stateful packet inspection firewall **reviews** the same packet information as a packet filtering firewall, but also **records** information about TCP connections.
- Some stateful firewalls also **keep track of TCP sequence numbers** to prevent attacks that depend on the sequence number, such as **session hijacking**.
- Some even **inspect** limited amounts of application data for some well-known protocols like **FTP, IM and SIP commands**, in order to identify and track related connections.

3. Application Level Gateway



3. Application Level Gateway

- An **application-level gateway**, also called an **application proxy**, acts as a relay of application-level traffic.
- The user contacts the gateway using a **TCP/IP application**, such as **Telnet or FTP**, and the gateway asks the user for the name of the remote host to be accessed.
- When the user responds and provides a **valid user ID** and **authentication information**, the gateway contacts the application on the remote host and relays TCP segments containing the application data between the two endpoints.

3. Application Level Gateway

- If the **gateway does not implement the proxy code for a specific application**, the service is not supported and cannot be forwarded across the firewall.
- Further, the gateway can be **configured to support only specific features** of an application that the network administrator considers acceptable while denying all other features.

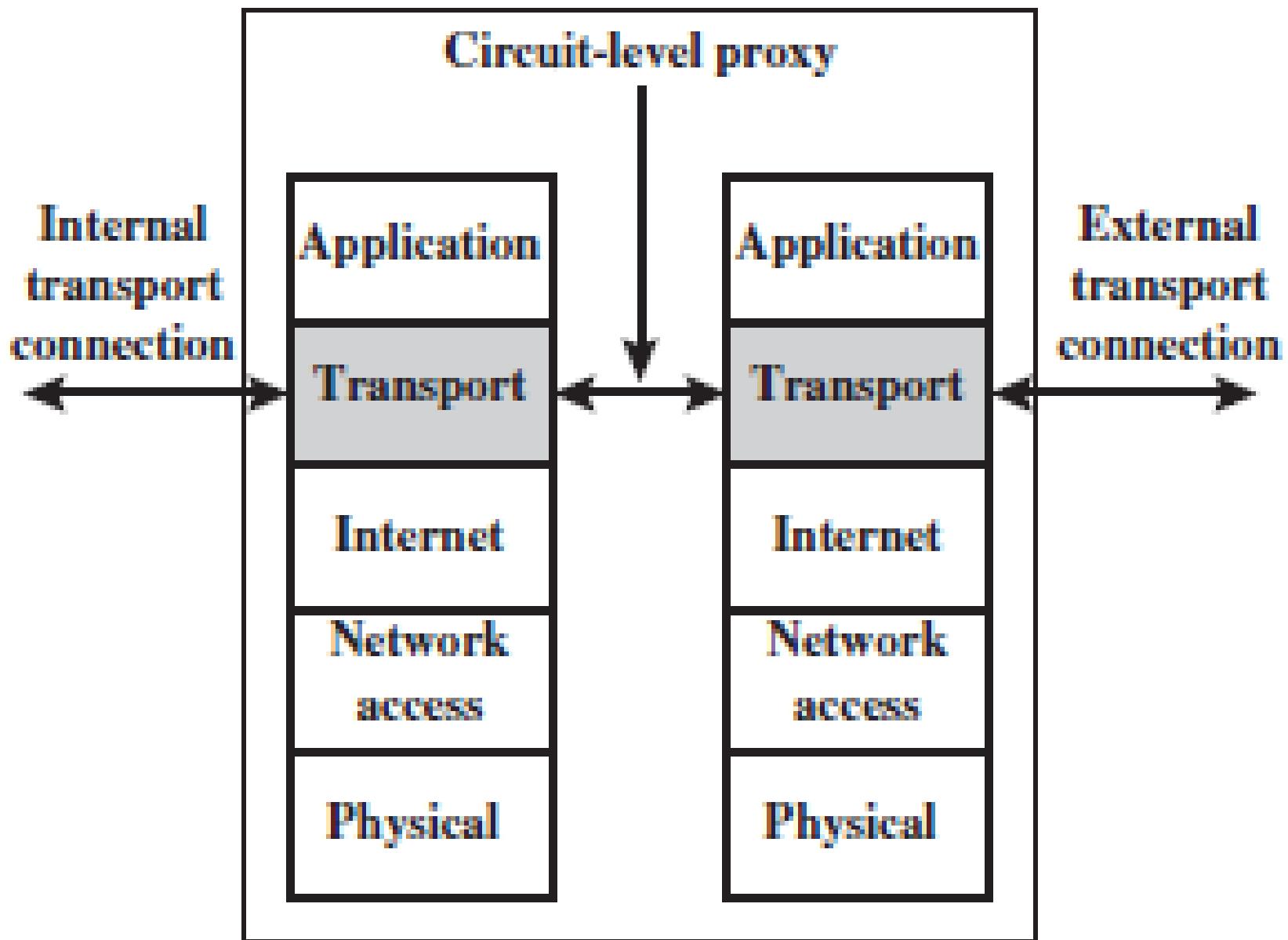
3. Application Level Gateway

- Application-level gateways tend to be more **secure than packet filters.**
- Rather than trying to deal with the numerous possible combinations that are to be allowed and forbidden at the TCP and IP level, the application-level gateway need only **scrutinize a few allowable applications.**
- In addition, it is **easy to log and audit** all incoming traffic at the application level.

3. Application Level Gateway

- A prime disadvantage of this type of gateway is the **additional processing overhead** on each connection.
- In effect, there are **two spliced connections** between the end users, with the gateway at the splice point, and the gateway must examine and forward all traffic in both directions.

4. Circuit Level Gateway



4. Circuit Level Gateway

- A fourth type of firewall is the circuit-level gateway or **circuit-level proxy**.
- This can be a **stand-alone system** or it can be a **specialized function performed by an application-level gateway**.
- As with an application gateway, a circuit-level gateway **does not permit an end-to-end TCP connection**.
- Rather, the gateway sets up **two TCP connections**, one between itself and a TCP user on an inner host and one between itself and a TCP user on an outside host.

4. Circuit Level Gateway

- Once the two connections are established, the gateway typically **relays TCP segments** from one connection to the other without examining the contents.
- The **security function** consists of determining which connections will be allowed.
- A typical use of circuit-level gateways is a situation in which the **system administrator trusts the internal users**.

4. Circuit Level Gateway

- The gateway can be configured to support **application-level** or proxy service on **inbound connections** and **circuit-level functions** for **outbound connections**.
- In this configuration, the gateway can incur the **processing overhead** of examining **incoming application data** for forbidden functions but does not incur that overhead on outgoing data.

SET

Secure Electronic Transactions

- An open encryption and security specification.
- Protect credit card transaction on the Internet.
- Companies involved:
 - MasterCard, Visa, IBM, Microsoft, Netscape, RSA, Terisa and Verisign
- Not a payment system.
- Set of security protocols and formats.

SET Services

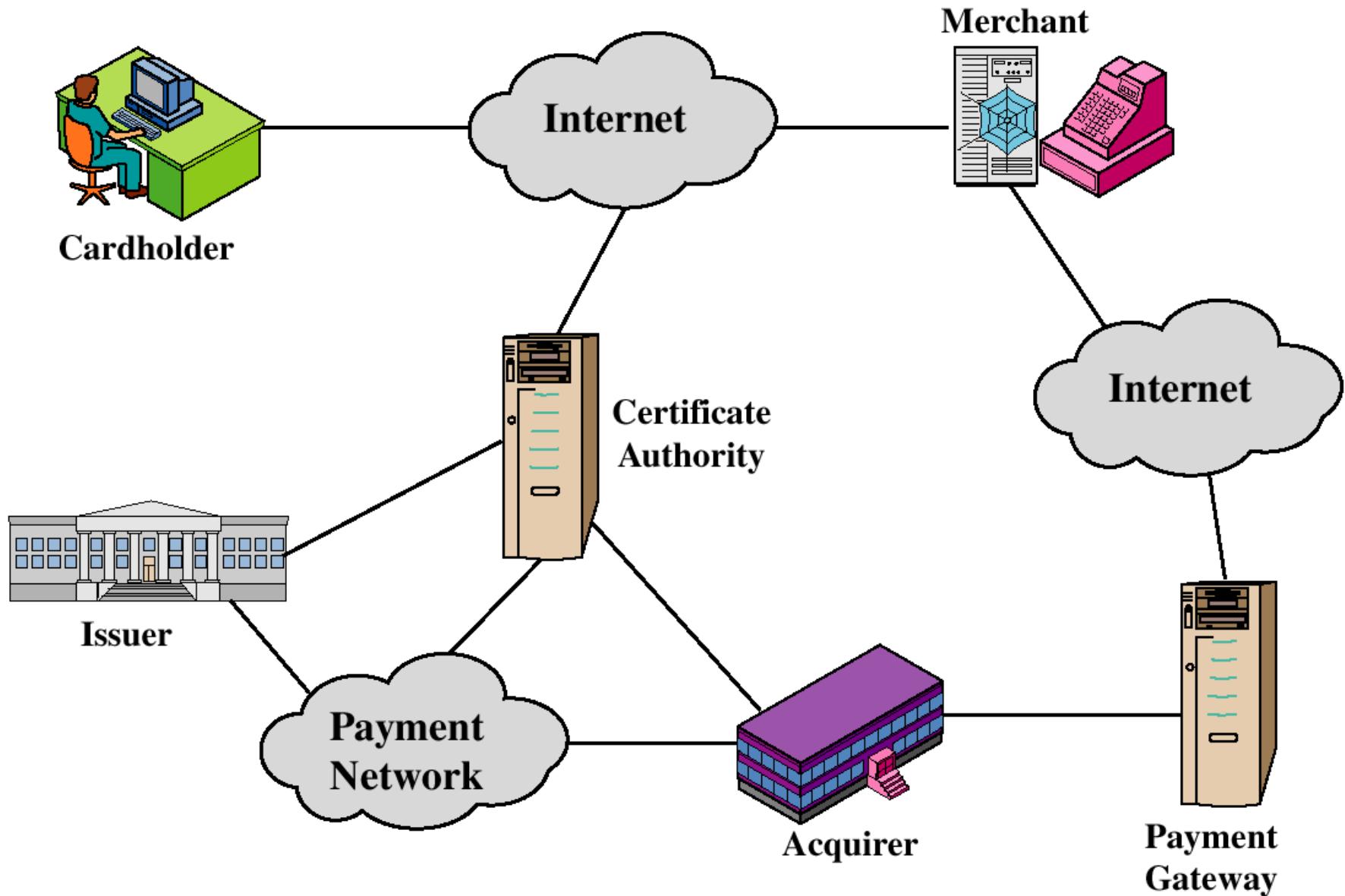
- Provides a secure communication channel in a transaction.
- Provides trust by the use of X.509v3 digital certificates.
- Ensures privacy.

SET Overview

- **Key Features of SET:**
 - Confidentiality of information
 - Integrity of data
 - Cardholder account authentication
 - Merchant authentication

www.rejinpaul.com

SET Participants



SET Participants

- **Card Holder:** Authorized holder of a payment card that has been issued by an issuer.
- **Merchant:** Person or organization that has goods or services to sell to the cardholder.
- **Issuer:** Financial institution (bank) that provides the cardholder with the payment card.
- **Acquirer:** Financial institution that establishes an account with a merchant and processes payment card authorizations and payments.
- **Payment Gateway:** Function operated by the acquirer or a designated third party that processes merchant payment messages.
- **Certification Authority:** An entity that is trusted to issue X.509v3 public key certificates for card holders, merchants and payment gateways.

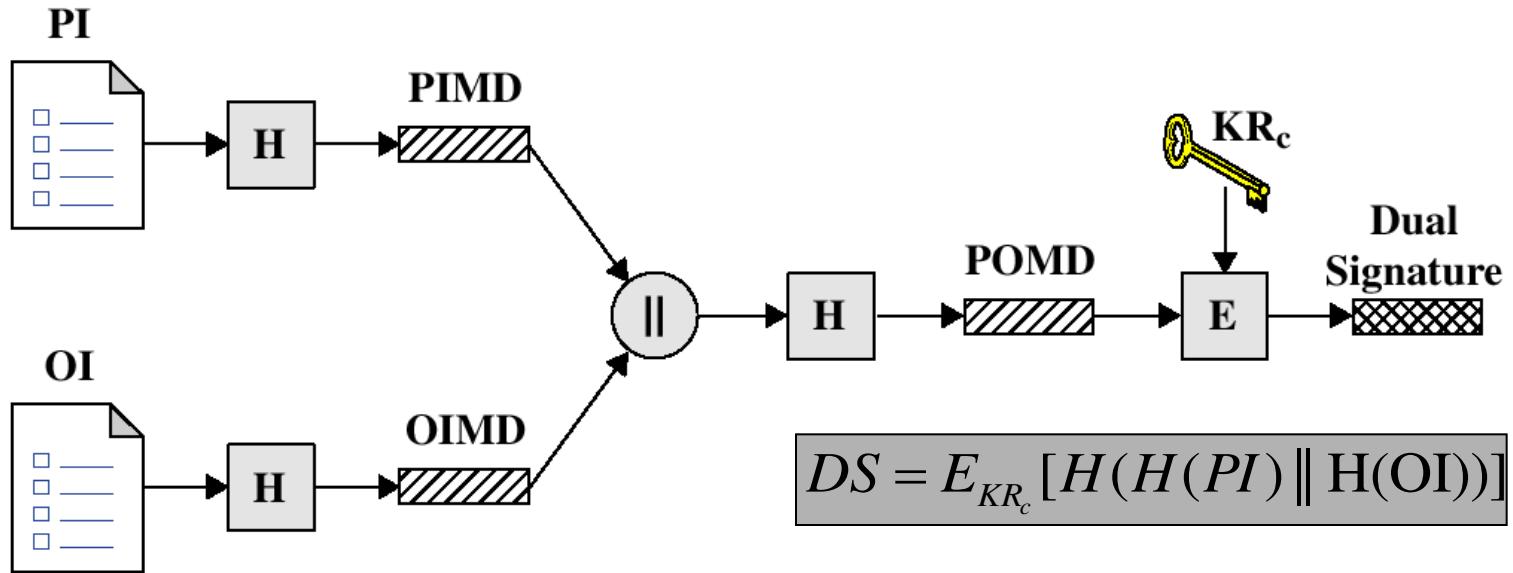
Sequence of events for transactions

1. **The customer opens an account:** Customer obtains a credit card account with a bank.
2. **The customer receives a certificate:** After verification the customer receives an X.509v3 digital certificate. The certificate verifies the customer's RSA public key.
3. **MERCHANTS HAVE THEIR OWN CERTIFICATES:** Merchant should possess 2 certificates for 2 public keys: one for signing messages and one for key exchange. He should also needs PG's public-key certificate.
4. **The customer places an order:** Customer browses through merchant's website to select items and determine price. He then sends a list of the items to be purchased to the merchant, who returns an order form containing list of items, their prices, total price and an order no.
5. **The merchant is verified:** In addition to the order form the merchant sends its certificate, so that the customer can verify that he is dealing with a valid store.

Sequence of events for transactions

6. **The order and payment are sent:** Customer sends the order and payment information to the merchant along with the customer's certificate. Order confirms the purchase and payment contains the credit card details. PI is encrypted that it cannot be read by the merchant. Customer's certificate enables the merchant to verify the customer.
7. **The merchant request payment authorization:** Merchant sends the PI to the PG requesting authorization that the customer's available credit is sufficient for this purchase.
8. **The merchant confirms the order:** Merchant sends confirmation of the order to the customer.
9. **The merchant provides the goods or service:** Merchant ships the goods to the customer.
10. **The merchant requests payments:** Request is sent to the PG which handles all of the payment processing.

Dual Signature



PI = Payment Information

OI = Order Information

H = Hash function (SHA-1)

\parallel = Concatenation

PIMD = PI message digest

OIMD = OI message digest

POMD = Payment Order message digest

E = Encryption (RSA)

KR_c = Customer's private signature key

Merchant Computes: $H(PIMD \parallel H(OI))$ and $D(PU_c, DS)$

If these two quantities are equal, then the merchant has verified the signature.

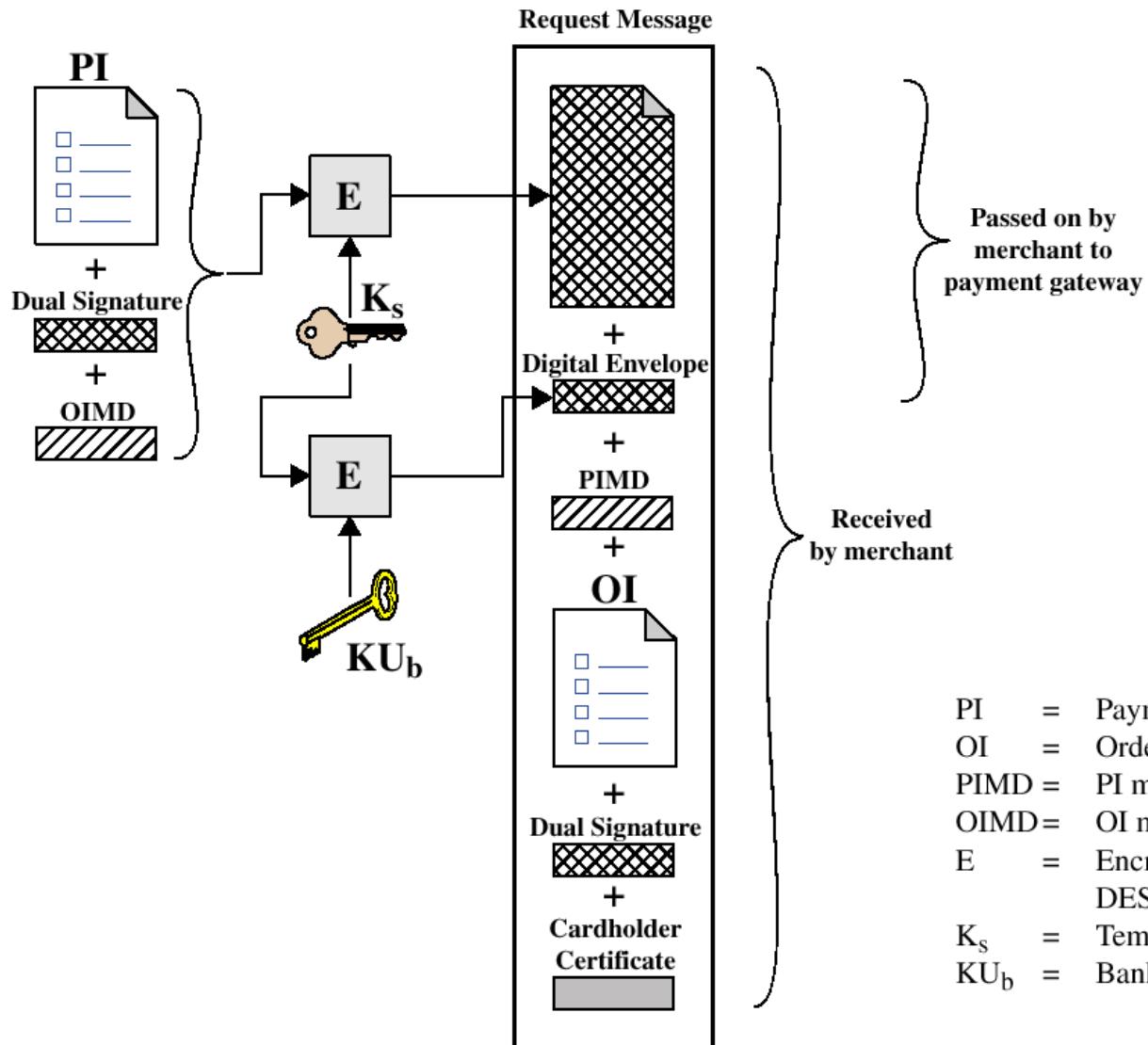
Bank Computes: $H(PI) \parallel OIMD$ and $D(PU_c, DS)$

If these two quantities are equal, then the bank has verified the signature.

Payment processing

- Purchase Request
 - Initiate Request – customer to merchant
 - Initiate Response – merchant's certificate and PG's certificate
 - Purchase Request – shown in the next slide
 - Purchase Response - acknowledgement
- Payment Authorization:
 - Authorization Request – merchant to PG
 - Authorization Response – PG to the merchant
- Payment Capture:
 - Capture Request – merchant to PG
 - Capture Response – PG to merchant (acquirer)

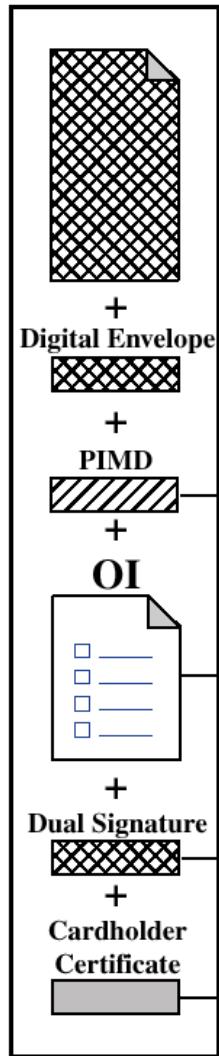
Payment processing



Cardholder sends Purchase Request

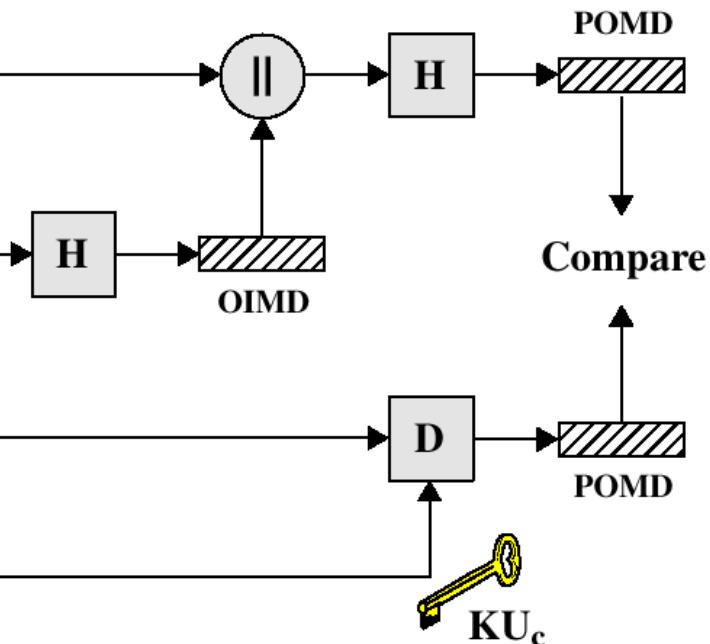
Payment processing

Request Message



Passed on by
merchant to
payment gateway

- | | | |
|-----------------|---|---------------------------------|
| OI | = | Order Information |
| OIMD | = | OI message digest |
| POMD | = | Payment Order message digest |
| D | = | Decryption (RSA) |
| H | = | Hash function (SHA-1) |
| KU _c | = | Customer's public signature key |



Merchant Verifies Customer Purchase Request

Intruders

Intruders

- **Three classes of intruders (hackers or crackers):**
 - **Masquerader**-an individual who is not authorized to use the computer and who penetrates a system's access controls to exploit a legitimate user's account
 - **Misfeasor**-a legitimate user who accesses data, programs or resources for which such access is not authorized
 - **Clandestine user**-an individual who seizes supervisory control of the system and uses this control to evade auditing and access controls or to suppress audit collection

Intrusion Techniques

- System maintain a file that associates a password with each authorized user.
- Password file can be protected with:
 - One-way encryption
 - Access Control

Intrusion Techniques

Techniques for guessing passwords:

- Try default passwords.
- Try all short words, 1 to 3 characters long.
- Try all the words in an electronic dictionary(60,000).
- Collect information about the user's hobbies, family names, birthday, etc.
- Try user's phone number, social security number, street address, etc.
- Try all license plate numbers (MUP103).
- Use a Trojan horse
- Tap the line between a remote user and the host system.

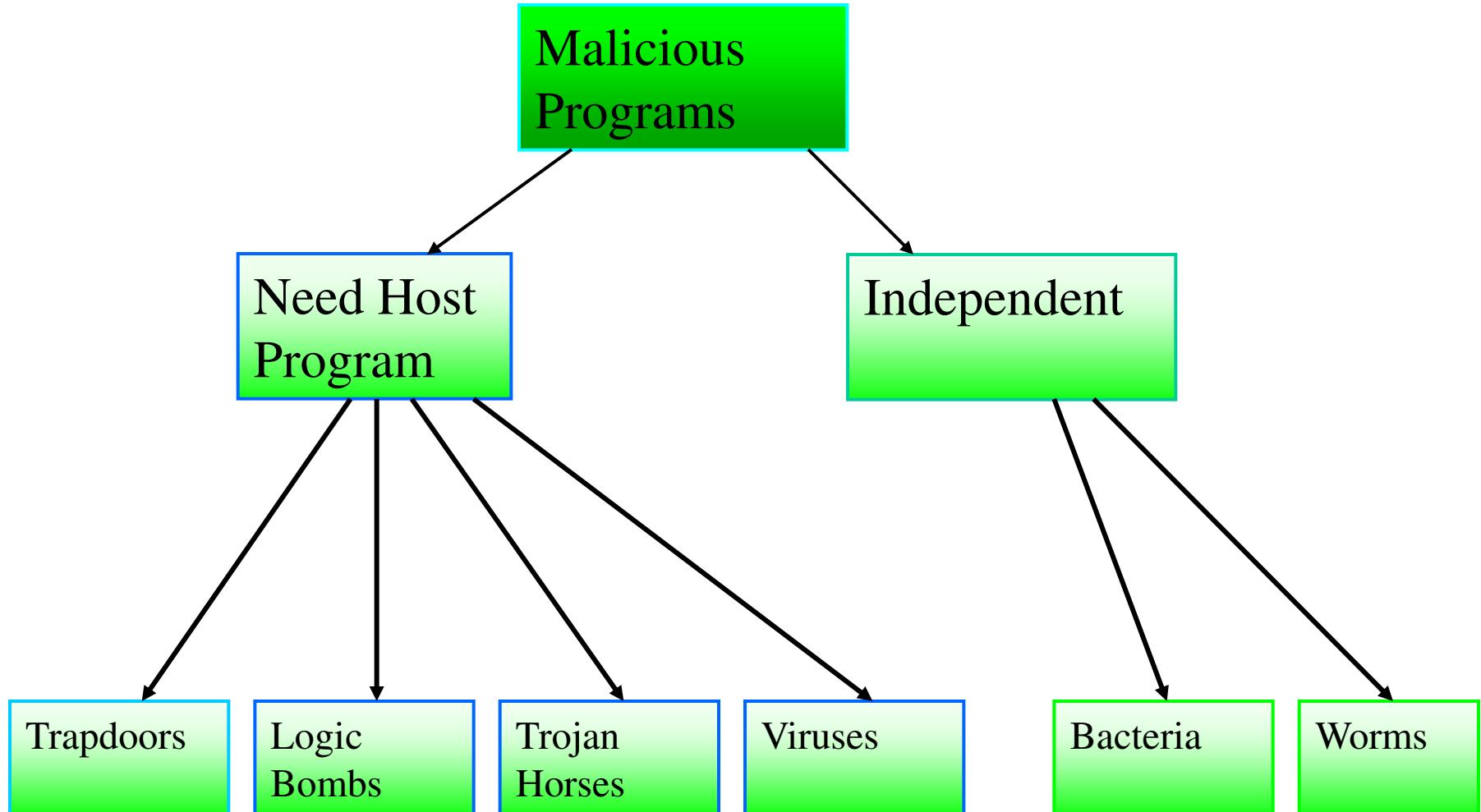
Prevention: Enforce good password selection (Ij4Gf4Se%of#)

Virus and related threats

Viruses and Malicious Programs

- Computer “Viruses” and related programs have the ability to replicate themselves on an ever increasing number of computers. They originally spread by people sharing floppy disks. Now they spread primarily over the Internet (a “Worm”).
- Other “Malicious Programs” may be installed by hand on a single machine. They may also be built into widely distributed commercial software packages. These are very hard to detect before the payload activates (Trojan Horses, Trap Doors, and Logic Bombs).

Taxanomy of Malicious Programs



Definitions

- A “Bacteria” replicates until it fills all disk space, or CPU cycles.
- Worm - a program that replicates itself across the network (usually riding on email messages or attached documents (e.g., macro viruses)).
- Virus - code that copies itself into other programs.
- Payload - harmful things the malicious program does, after it has had time to spread.

Definitions

- Trojan Horse - instructions in an otherwise good program that cause bad things to happen (sending your data or password to an attacker over the net).
- Logic Bomb - malicious code that activates on an event (e.g., date).
- Trap Door (or Back Door) - undocumented entry point written into code for debugging that can allow unwanted users.
- Easter Egg - extraneous code that does something “cool.” A way for programmers to show that they control the product.

Virus Phases

- **Dormant phase** - the virus is idle, will be activated by some events like date
- **Propagation phase** - the virus places an identical copy of itself into other programs
- **Triggering phase** – the virus is activated to perform the function for which it was intended
- **Execution phase** – the function is performed

Virus Protection

Have a well-known virus protection program, configured to scan disks and downloads automatically for known viruses.

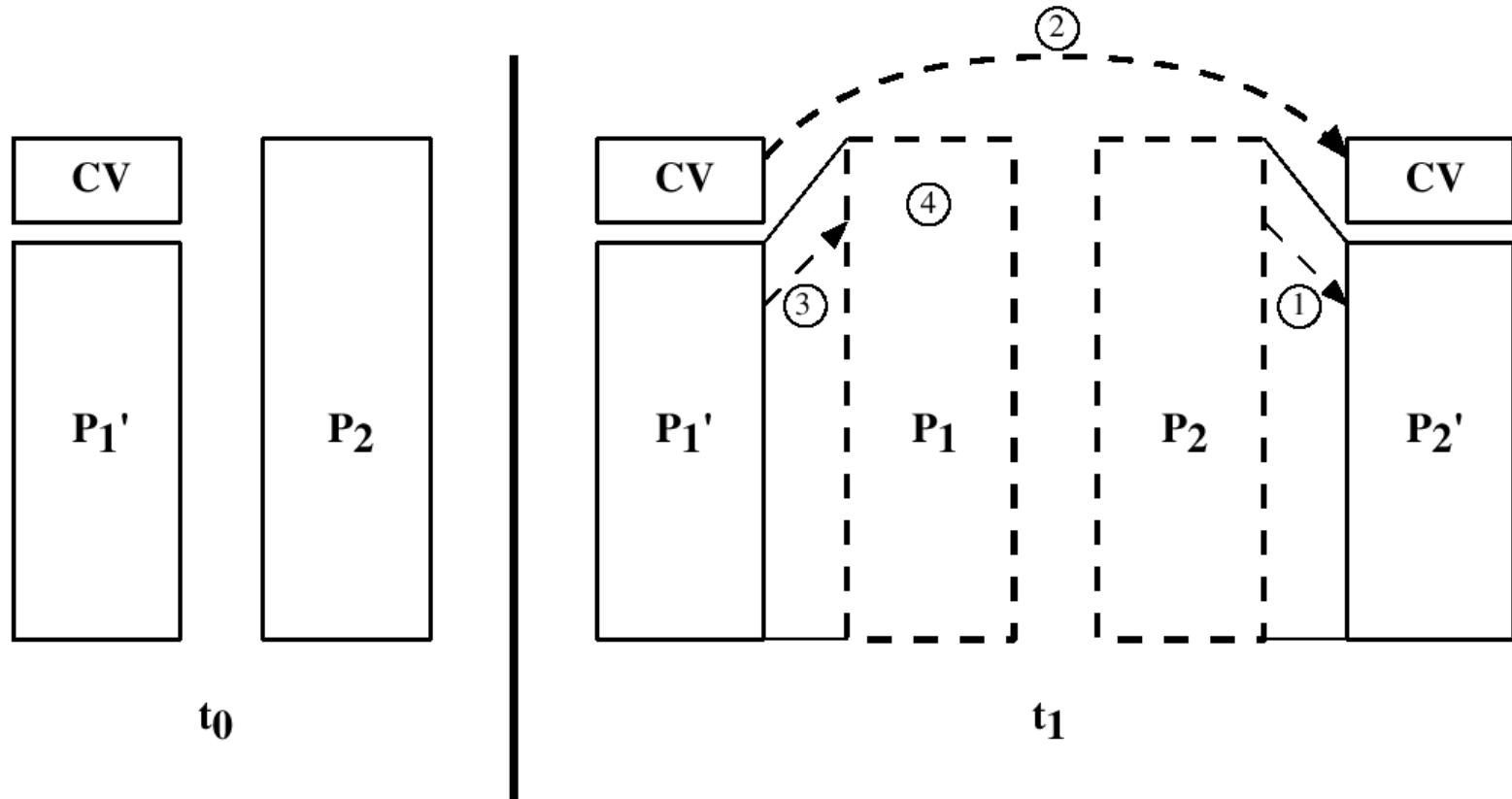
Do not execute programs (or "macro's") from unknown sources (e.g., PS files, Hypercard files, MS Office documents,

Avoid the most common operating systems and email programs, if possible.

Virus Structure

```
program V :=  
  
    {goto main;  
     1234567;  
  
     subroutine infect-executable :=  
         {loop:  
          file := get-random-executable-file;  
          if (first-line-of-file = 1234567)  
              then goto loop  
              else prepend V to file; }  
  
     subroutine do-damage :=  
         {whatever damage is to be done}  
  
     subroutine trigger-pulled :=  
         {return true if some condition holds}  
  
main:   main-program :=  
        {infect-executable;  
         if trigger-pulled then do-damage;  
         goto next;}  
  
next:  
    }
```

A Compression Virus



Types of Viruses

- **Parasitic Virus** - attaches itself to executable files as part of their code and replicates. Runs whenever the host program runs.
- **Memory-resident Virus** - Lodges in main memory as part of the residual operating system.
- **Boot Sector Virus** - infects the boot sector of a disk, and spreads when the operating system boots up (DOS viruses).
- **Stealth Virus** - explicitly designed to hide from Virus Scanning programs.
- **Polymorphic Virus** - mutates with every new host to prevent signature detection.
- **Metamorphic Virus** – mutates with every infection. Difference is it rewrites itself completely

Macro Viruses

- Microsoft Office applications allow “macros” to be part of the document. The macro could run whenever the document is opened, or when a certain command is selected (Save File).
- Platform independent.
- Infect documents, delete files, generate email and edit letters.
- Easily spread, common method is emails

Email Viruses

- Email viruses sends itself to everyone in the mailing list
- Does local damage

Antivirus Approaches

Goals: Detection
Identification
Removal

1st Generation, Scanners: searched files for any of a library of known virus “signatures.” Checked executable files for length changes.

2nd Generation, Heuristic Scanners: looks for more general signs than specific signatures (code segments common to many viruses).
Checked files for checksum or hash changes.

3rd Generation, Activity Traps: stay resident in memory and look for certain patterns of software behavior (e.g., scanning files).

4th Generation, Full Featured: combine the best of the techniques above.

Advanced Antivirus Techniques

- **Generic Decryption (GD)**
 - **CPU Emulator** - A s/w based virtual computer.
Instructions in an exe file are interpreted rather than executed in the processor
 - **Virus Signature Scanner** - Module that scans the target code looking for known signatures
 - **Emulation Control Module** - controls the execution of the target code

Advanced Antivirus Techniques

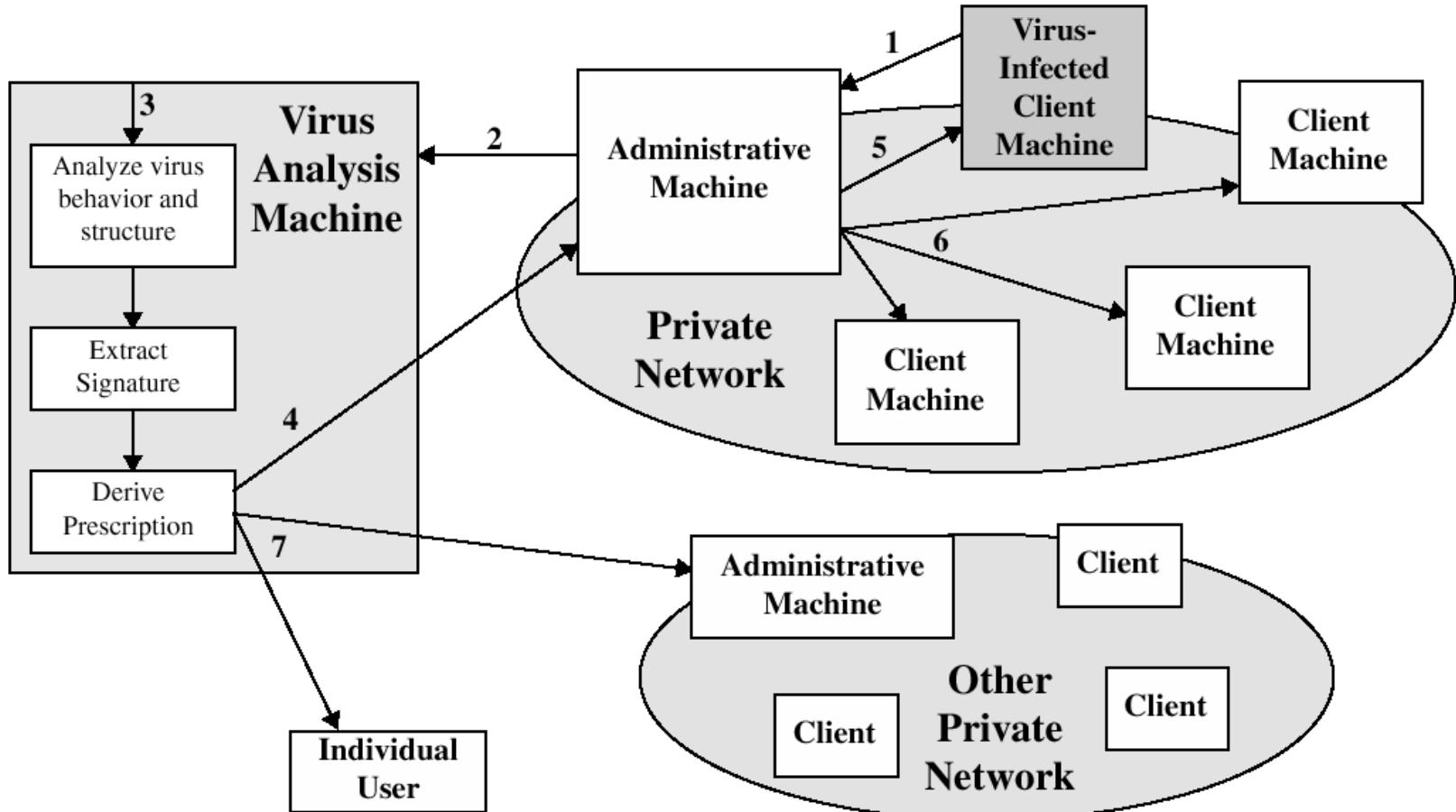


Figure 9.11 Digital Immune System

Trusted Systems

Trusted Computer Systems

- Information system security is increasingly important
- Have varying degrees of sensitivity of information
- Subjects (people or programs) have varying rights of access to objects (information)

Hence:

Information system security is the application of managerial and administrative procedures and technical and physical safeguards to ensure not only the **confidentiality, integrity** and **availability** of information which is processed by an information system, but also of the information system itself, together with its environment.

Types of Secure Computing Systems

- **Dedicated (Single-Level) Systems**
 - Handles subjects and objects with same classification
 - Relies on other security procedures (eg physical)
- **System-High**
 - Only provides need-to-know protection between users
 - Entire system operates at highest classification level
 - All users must be cleared for that level of information
- **Compartmented**
 - Variation of System-High which can process two or more types of compartmented information
 - Not all users are cleared for all compartments, but all must be cleared to the highest level of information processed

Types of Secure Computing Systems

- **Multi-Level Systems**

- Is validated for handling subjects and objects with different rights and levels of security simultaneously
- **Major features of such systems include:**
 - user identification and authentication
 - resource access control and object labelling
 - audit trails of all security relevant events
 - external validation of the systems security

Network Security



Dr. B. Shanthini
Professor and Head
St. Peter's College of Engineering & Technology 1

Unit 5 - Outline

E-mail Security: Security Services for E-mail-attacks possible through E-mail - establishing keys privacy-authentication of the source-Message Integrity-Non-repudiation-Pretty Good Privacy-S/MIME.

IPSecurity: Overview of IPSec - IP and IPv6-Authentication Header-ESP-Internet Key Exchange (Phases of IKE, ISAKMP/IKE Encoding).

Web Security: SSL/TLS Basic Protocol-computing the keys- client authentication-PKI as deployed by SSL Attacks fixed in v3-Exportability-Encoding-Secure Electronic Transaction (SET).

Email Security

Email Security Enhancements

- Confidentiality
 - protection from disclosure
- Authentication
 - of sender of message
- Message integrity
 - protection from modification
- Non-repudiation of origin
 - protection from denial by sender

Pretty Good Privacy (PGP)

- Widely used de facto secure email
- Developed by Phil Zimmermann
- Selected best available crypto algs to use
- Integrated into a single program
- On Unix, PC, Macintosh and other systems
- Originally free, now also have commercial versions available

PGP Operation

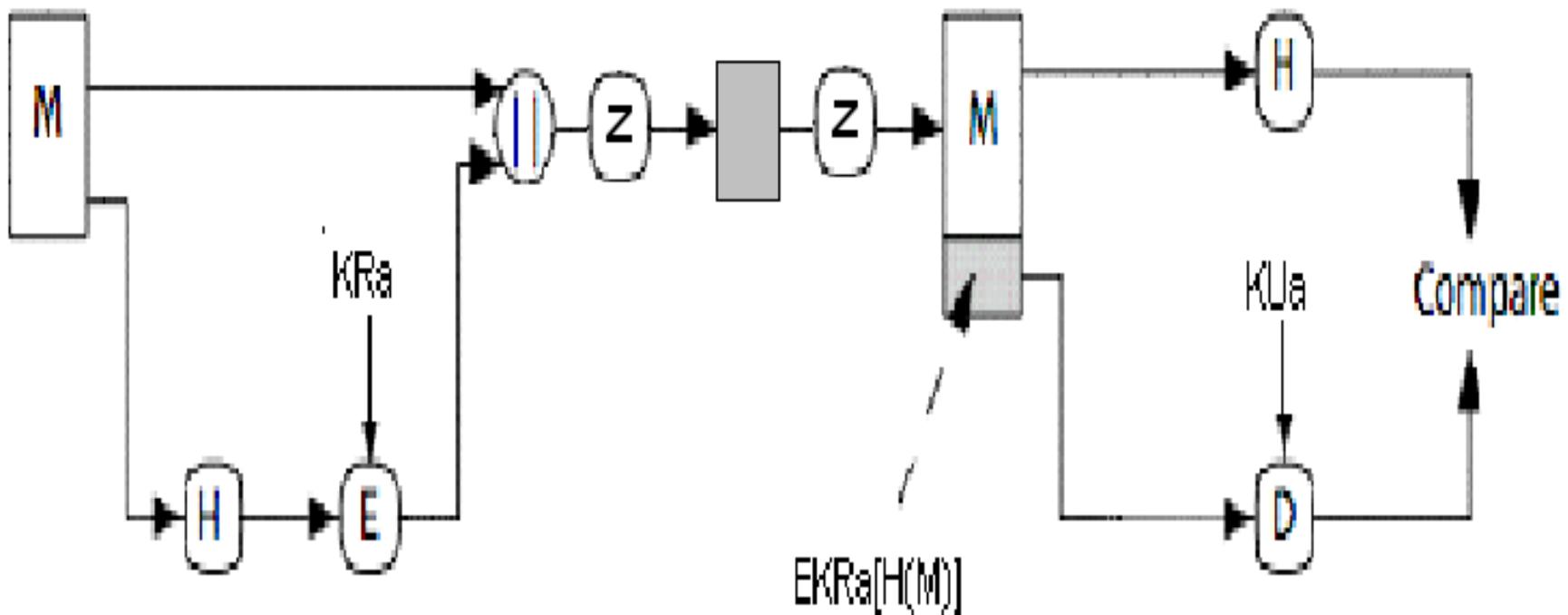
Consists of 5 services:

- Authentication
- Confidentiality
- Compression
- E-mail Compatibility
- Segmentation

PGP Operation - Authentication

1. Sender creates message
2. Use SHA-1 to generate 160-bit hash of message
3. Hash code is encrypted with RSA using sender's private key, and is attached to message
4. Receiver uses RSA with sender's public key to decrypt and recover hash code
5. Receiver generates a new hash code for the received message and compares it with the decrypted hash code
6. If the two match, the message is accepted

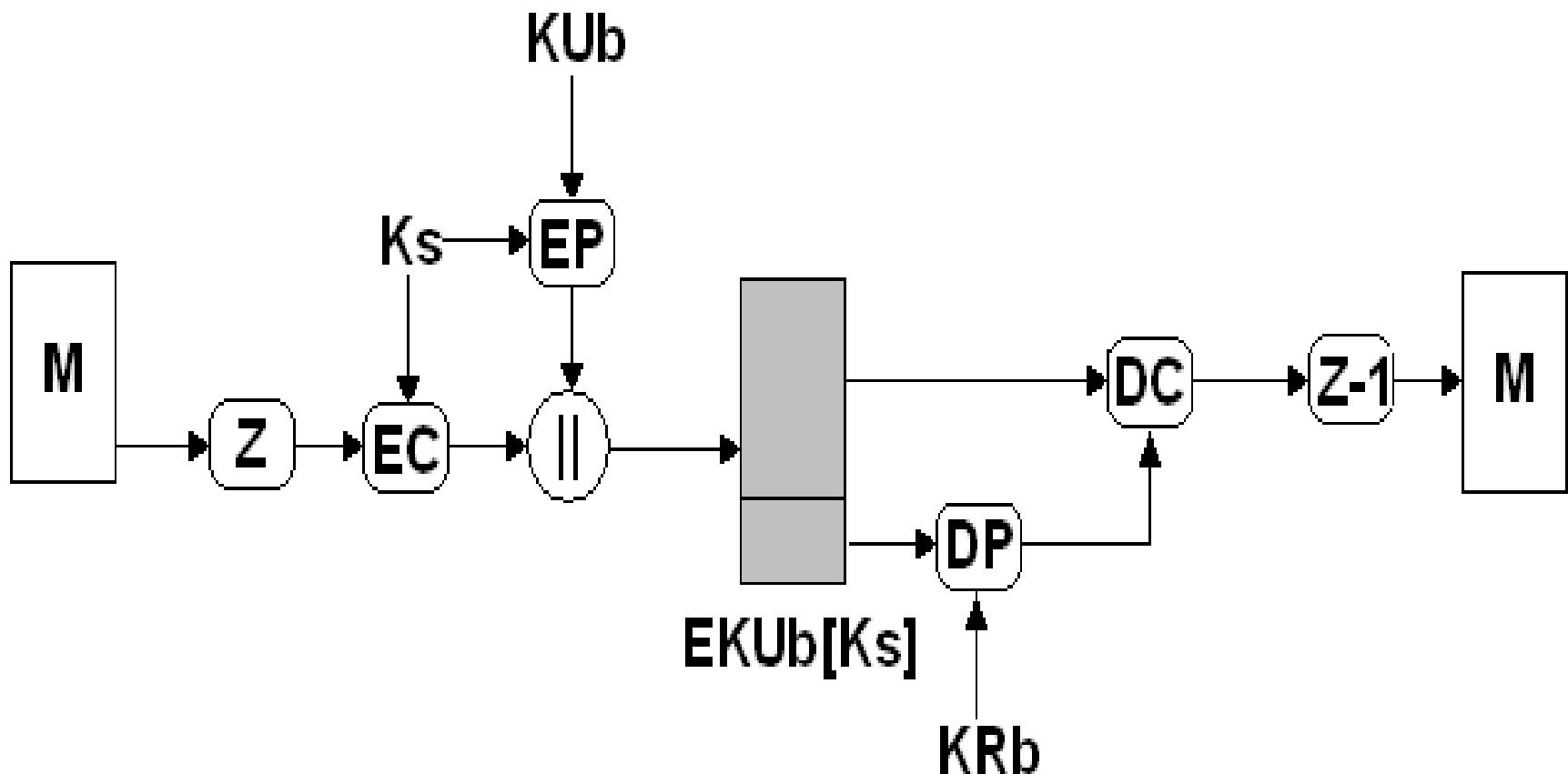
Authentication Only



PGP Operation - Confidentiality

1. Sender generates message and 128-bit random number as session key for it
2. Encrypt message using CAST-128 / IDEA / 3DES in CBC mode with session key
3. Session key encrypted using RSA with recipient's public key & attached to msg
4. Receiver uses RSA with private key to decrypt and recover session key
5. Session key is used to decrypt message

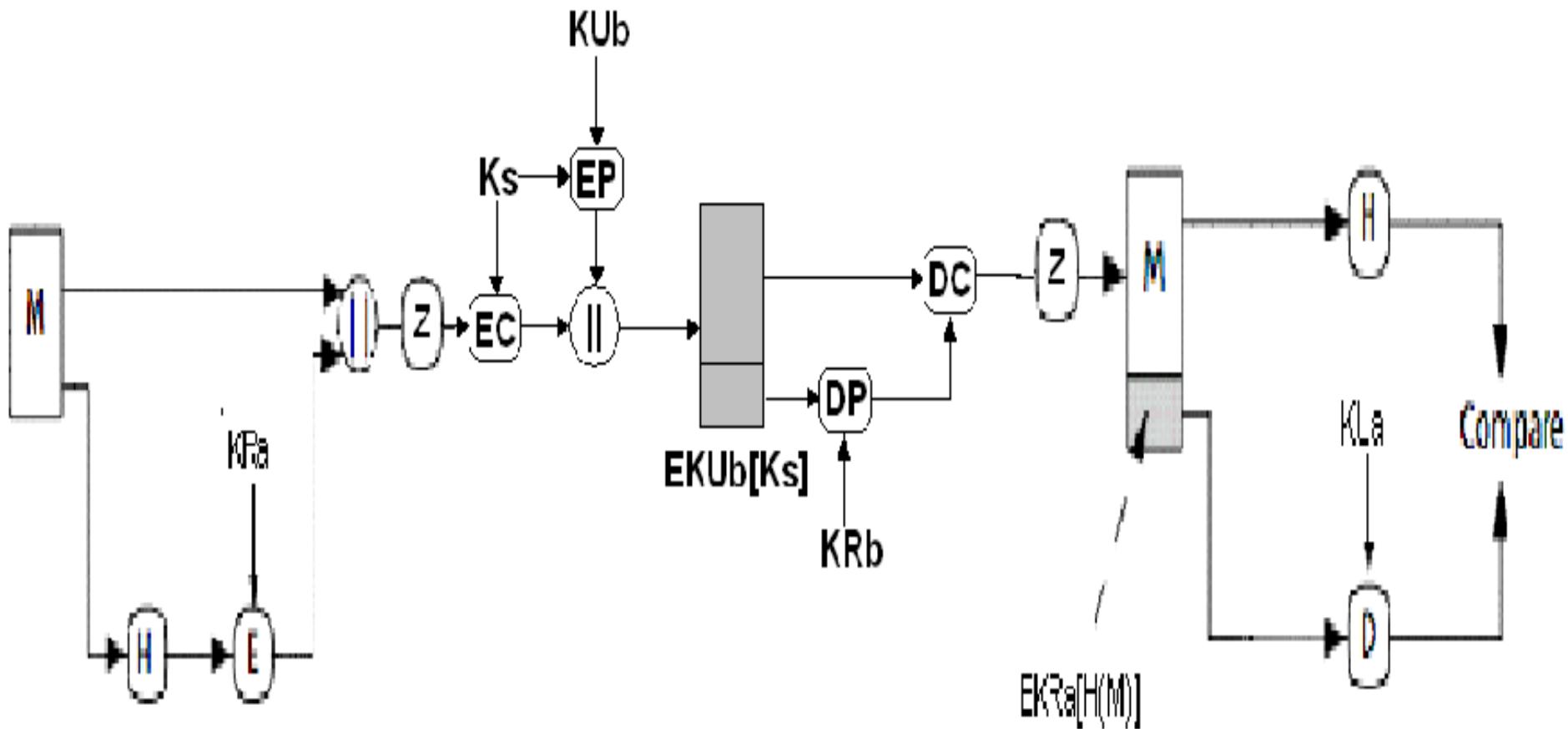
Confidentiality Only



PGP Operation - Confidentiality & Authentication

- Can use both services on same message
 - Create signature & attach to message
 - Encrypt both message & signature using CAST-128 (or IDEA or 3DES)
 - Session key is encrypted using RSA (or ElGamal)

Confidentiality & Authentication



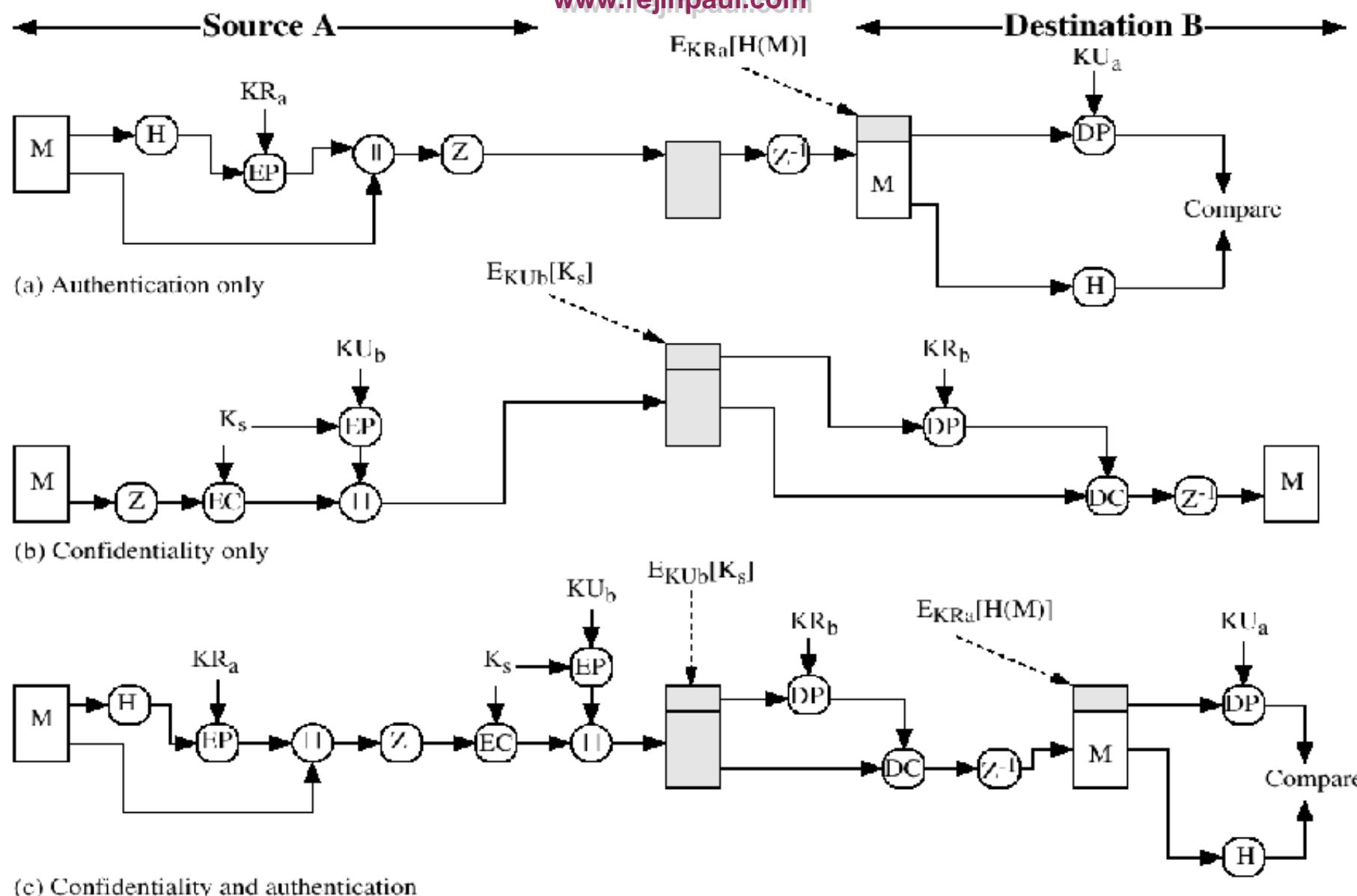


Figure 5.1 PGP Cryptographic Functions

PGP Operation - Compression

- By default PGP compresses message after signing but before encrypting
 - So we can store uncompressed message & signature for later verification
- Uses ZIP compression algorithm

PGP Operation – Email Compatibility

- When using PGP will have binary data to send (encrypted message etc)
- However email was designed only for text
- Hence PGP must encode raw binary data into printable ASCII characters
- Uses radix-64 algorithm
 - maps 3 bytes to 4 printable chars
 - also appends a CRC
- PGP also segments messages if too big

E-mail Compatibility

- The scheme used is radix-64 conversion
- The use of radix-64 expands the message by 33%.

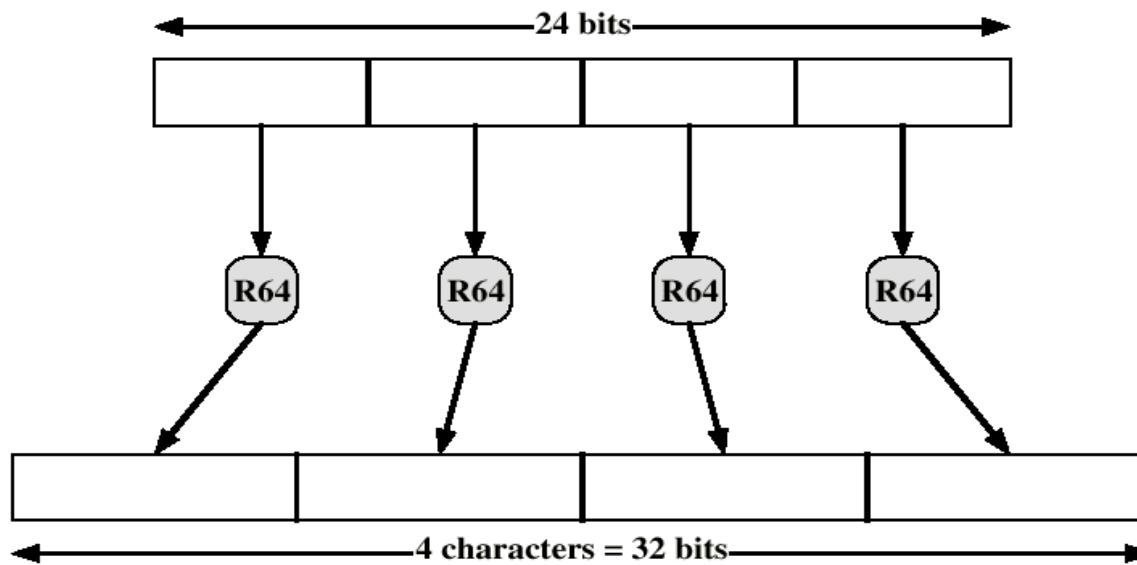
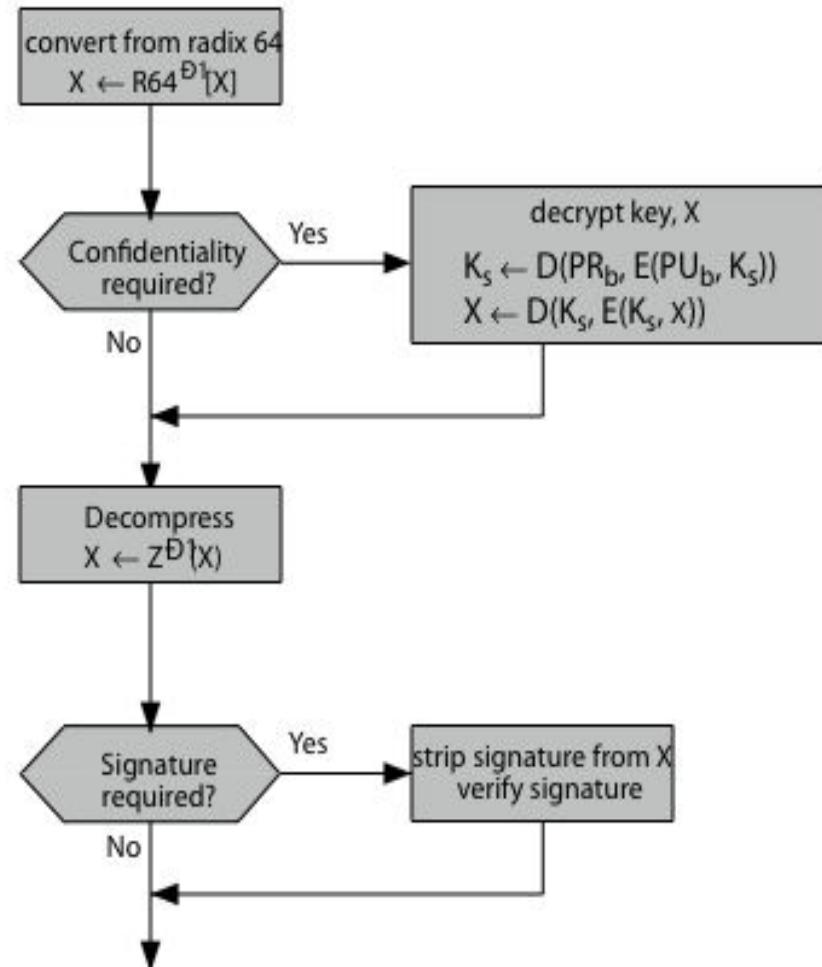
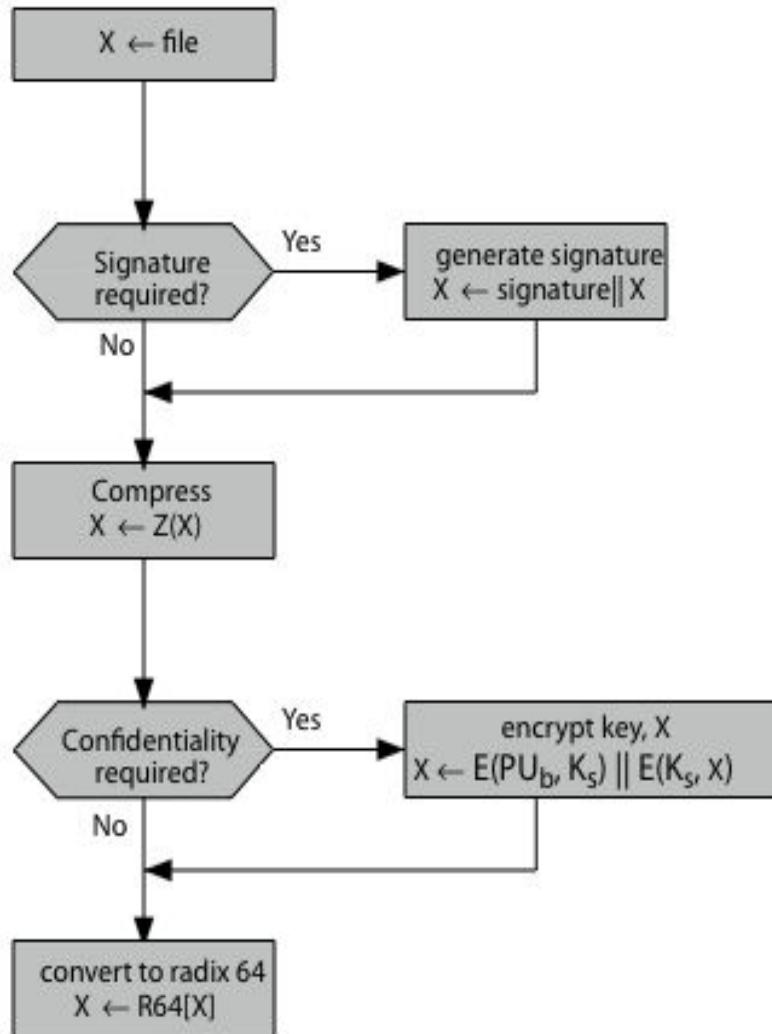


Figure 5.11 Printable Encoding of Binary Data into Radix-64 Format

PGP Operation – Segmentation & Reassembly

- E-mail facilities are restricted to a maximum length of 50,000 bytes.
- Any message longer than that must be broken into smaller segments
- Segmentation is done after all the processing
- At the receiving end, PGP strip off all e-mail headers and reassemble the entire original block

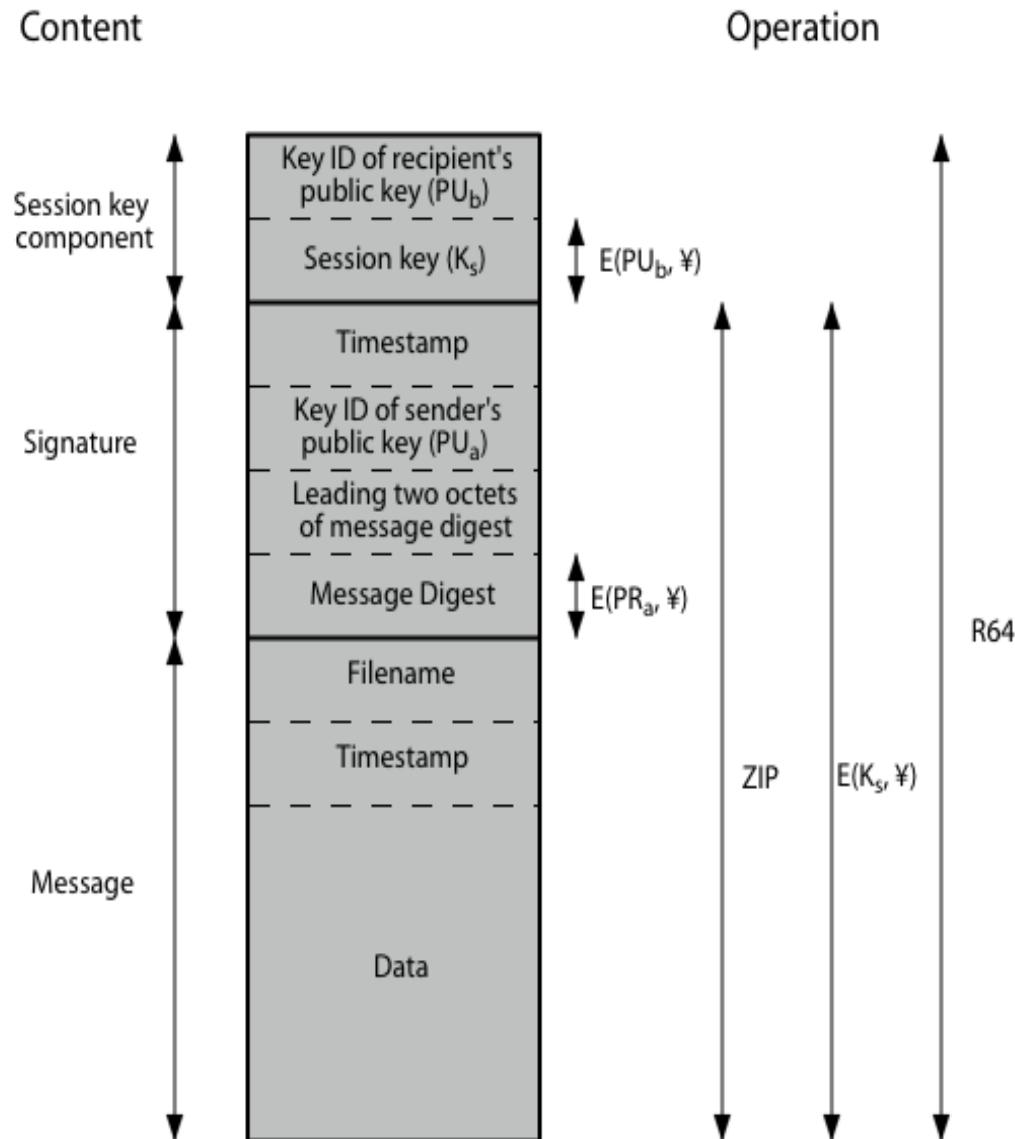
PGP Operation - Summary



(a) Generic Transmission Diagram (from A)

(b) Generic Reception Diagram (to B)

PGP Message Format



PGP Key Rings

- Each PGP user has a pair of key rings:
 - public-key ring contains all the public-keys of other PGP users known to this user, indexed by key ID
 - private-key ring contains the public/private key pair(s) for this user, indexed by key ID

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

Private Key Ring

- **Timestamp:** The date/time when this key pair was generated.
- **Key ID:** The least significant 64 bits of the public key for this entry.
- **Public key:** The public-key portion of the pair.
- **Private key:** The private-key portion of the pair; this field is encrypted.
- **User ID:** Typically, this will be the user's e-mail address (e.g., stallings@acm.org).

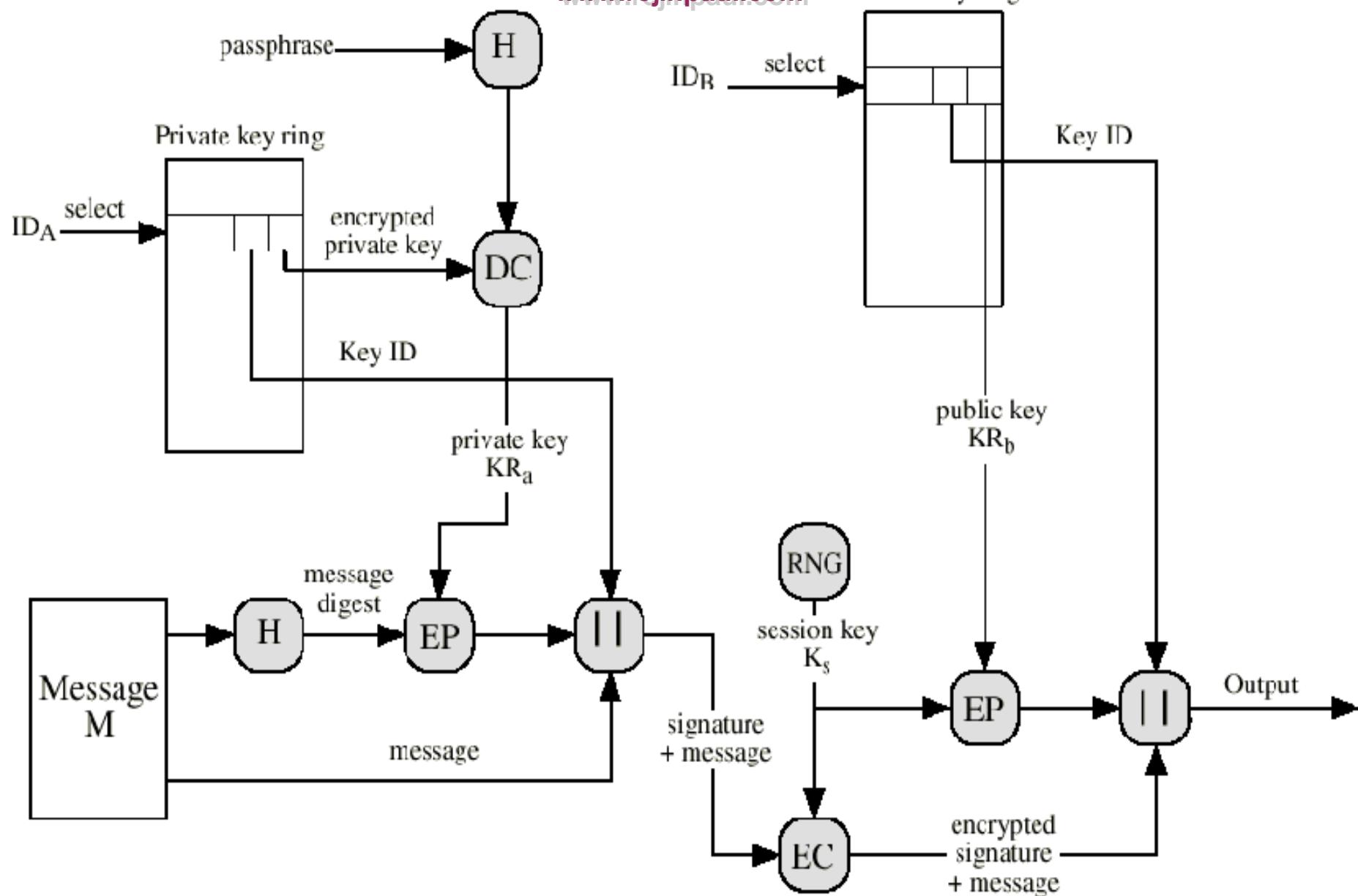


Figure 5.5 PGP Message Generation (from User A to User B; no compression or radix 64 conversion)

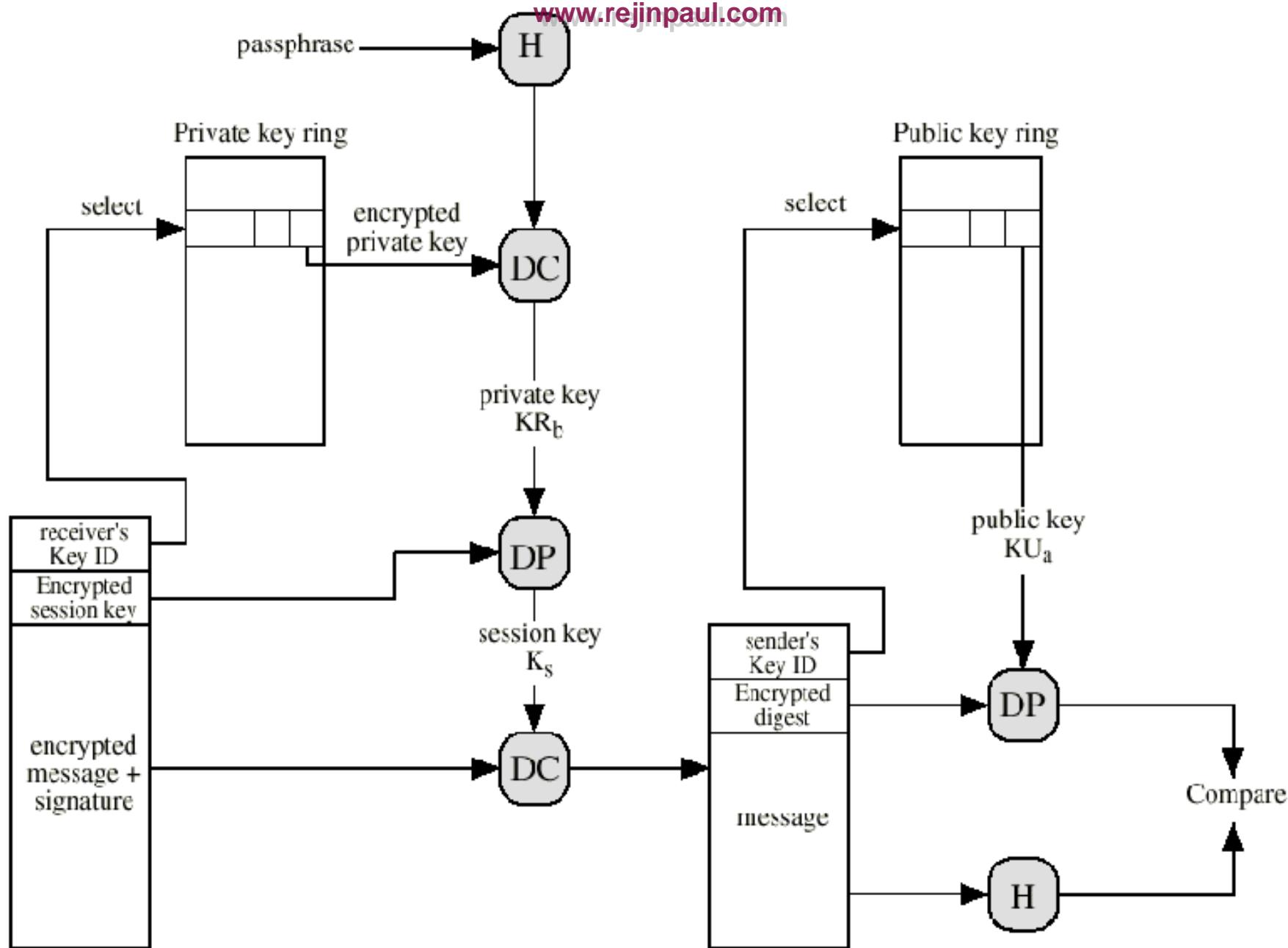


Figure 5.6 PGP Message Reception (from User A to User B; no compression or radix 64 conversion)

Summary of PGP Services

Function	Algorithm Used
Digital Signature	DSS/SHA or RSA/SHA
Message Encryption	CAST or IDEA or three-key triple DES with Diffie-Hellman or RSA
Compression	ZIP
E-mail Compatibility	Radix-64 conversion
Segmentation	-

S/MIME

(Secure/Multipurpose Internet Mail Extensions)

- Security enhancement to MIME email
 - Original Internet RFC822 email was text only
 - MIME provided support for varying content types & multi-part messages with encoding of binary data to textual form
 - S/MIME adds security enhancements
- Have S/MIME support in many mail agents
 - eg MS Outlook, Mozilla, Mac Mail etc

RFC 822

- Defines the format for the text messages that are sent through e-mail
- In RFC 822 messages have envelope and contents
- Envelope contains information needed to accomplish transmission and delivery
- Contents compose the object to be delivered to the recipient
- Message contains some header lines followed by unrestricted text
- Header is separated by the body by a blank line
- Header line consists of a keyword, followed by a colon, followed by the keyword's arguments
- Frequently used keywords are From, To, Subject and Date
- Ex. From : "William Stallings" <ws@shore.net>

(Multipurpose Internet Mail Extensions)

- MIME is the extension of RFC 822 framework
- Used to address the limitations of SMTP / RFC 822

Limitations of SMTP / RFC 822:

1. SMTP cannot send executable files or binary files
2. SMTP cannot transmit text data that includes national language characters (8 bits) but SMTP limited to 7-bit ASCII
3. SMTP servers may reject mails over a certain size
4. SMTP gateways have translation problems

Limitations of SMTP / RFC

822: (Contd...)

5. SMTP gateways to X.400 e-mail networks cannot handle non-textual data
6. Some SMTP implementations has the following problems:
 - Deletion, addition or reordering of carriage return and linefeed
 - Truncating and wrapping lines longer than 76 chars
 - Removal of trailing white spaces (tab & space char)
 - Padding of lines in a message to the same length
 - Conversion of Tab characters into multiple space chars

MIME Overview

MIME spec. includes the following elements:

1. Five new message headers are defined
2. A number of content formats are defined
3. Transfer Encodings are defined

Header fields in MIME

- **MIME-Version:** Must be "1.0"
- **Content-Type:** More types being added by developers (application/word)
- **Content-Transfer-Encoding:** How message has been encoded
- **Content-ID:** Used to identify MIME entities uniquely in multiple contents
- **Content Description:** Needed when content is not readable text (e.g. mpeg)

MIME Content Types

Type	Subtype
Text	Plain Enriched
Multipart	Mixed Parallel Alternative Digest
Message	rfc 822 Partial External Body

MIME Content Types - Continued

Type	Subtype
Image	jpeg gif
Video	mpeg
Audio	Basic
Application	Post Script Octet Stream

MIME Transfer Encodings

7 bit	ASCII Characters
8 bit	Non-ASCII Characters may also present
Binary	Non-ASCII & not short enough for SMTP transport
Quoted-Printable	Encoded to ASCII text
Base64	Encodes data by mapping 6-bits to 8-bits which are printable ASCII characters
X-token	A named nonstandard encoding

S/MIME Cryptographic Algorithms

- Digital signatures: DSS & RSA
- Hash functions: SHA-1 & MD5
- Session key encryption: ElGamal & RSA
- Message encryption: AES, Triple-DES, RC2/40 and others
- MAC: HMAC with SHA-1
- Have process to decide which algs to use

S/MIME Functions

- Enveloped data
 - Encrypted content and associated keys
- Signed data
 - Encoded message + signed digest
- Clear-signed data
 - Clear text message + encoded signed digest
- Signed & enveloped data
 - Nesting of signed & encrypted entities

Enveloped Data

Steps for preparing an Enveloped Data:

- Generate a pseudorandom session key for a particular symmetric encryption algorithm (RC2/40 or Triple DES)
- For each recipient, encrypt the session key with the recipient's public RSA key and share it with them.
- For each recipient, prepare a block **RecipientInfo** that contains an identifier of the recipient's public-key certificate, an identifier of the algorithm used to encrypt the session key and the encrypted session key.
- Encrypt the message content with the session key.

Signed Data

Steps for preparing an Signed Data:

- Select a message digest algorithm (SHA or MD5)
- Compute the message digest, or hash function of the content to be signed.
- Encrypt the message digest with the signer's private key.
- Prepare a block, **SignerInfo** that contains the signer's public-key certificate, an identifier of the message digest algorithm, an identifier of the algorithm used to encrypt the message digest and the encrypted message digest.

Clear Signed Data

- Clear signing is achieved using multipart content type with the signed sub type.
- Signing process does not involve transforming the message to be signed, so that the message is sent "in the clear".
- Thus the recipients with MIME capability but not S/MIME capability are able to read the incoming message.

Registration Request

- An application or user will apply to a certification authority for a public-key certificate.
- A Certification Request including **certificationRequestInfo** block, an identifier of the public-key encryption algorithm, the signature of the **certificationRequestInfo** block, a name of the certificate subject and the user's public key is sent using application/pkcs10.

Certificates-only Message

- A message containing only certificates or certificate revocation list (CRL) can be sent in response to a registration request.
- Message is an application/pkcs7-MIME type/Subtype with an smime-type parameter of degenerate.
- Steps involved are the same as those for signedData message, except there is no message content and the signerInfo field is empty.

S/MIME Certificate Processing

- S/MIME uses X.509 v3 certificates.
- Managed using a hybrid of a strict X.509 CA hierarchy & PGP's web of trust.
- Each client has a list of trusted CA's certs and own public/private key pairs & certs.
- Certificates must be signed by trusted CA's.

Certificate Authorities

- Have several well-known CA's
- Verisign one of most widely used
- Verisign issues several types of Digital IDs
- increasing levels of checks & hence trust

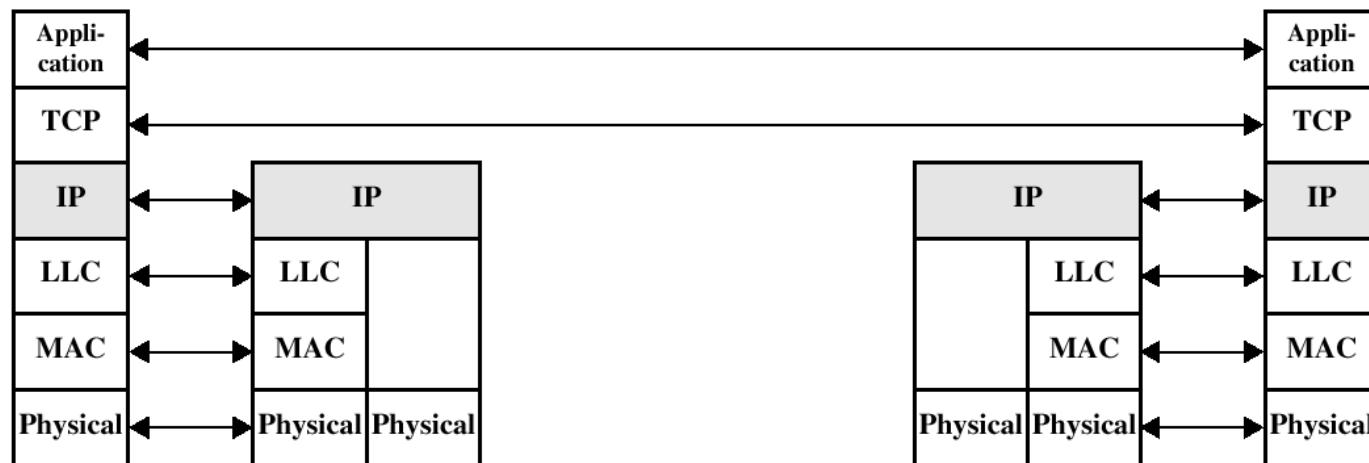
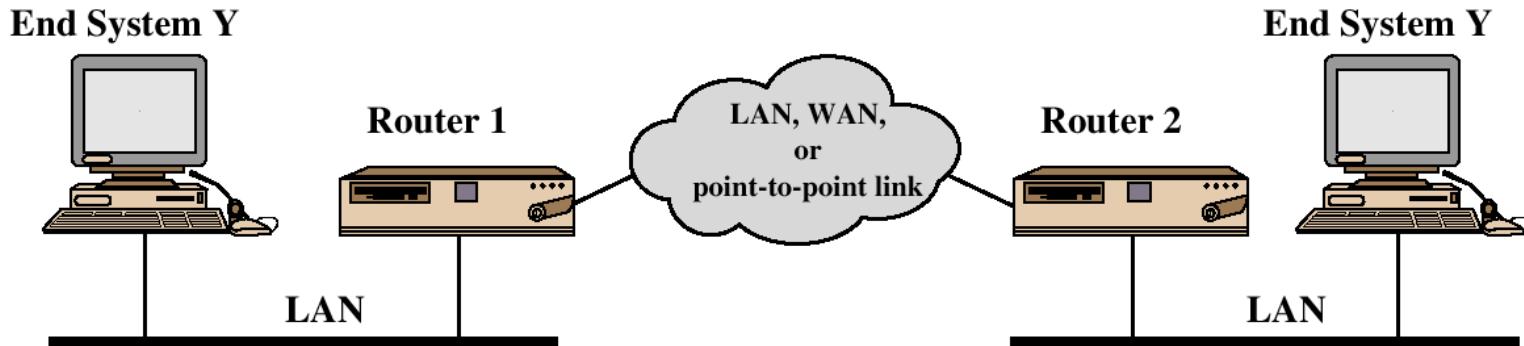
Class	Identity Checks	Usage
1	name/email check	web browsing/email
2	+ enroll/addr check	email, subs, s/w validate
3	+ ID documents	e-banking/service access

IP Security

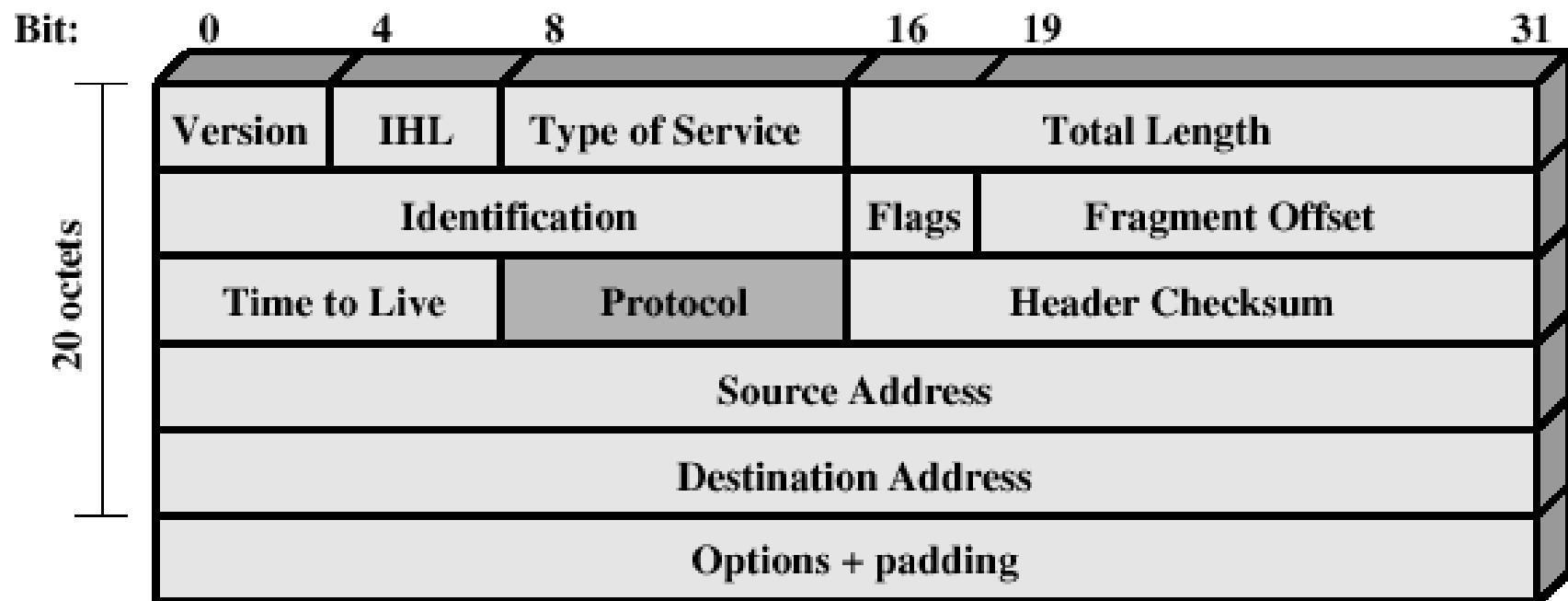
Outline

- Internetworking and Internet Protocols
- IP Security Overview
- IP Security Architecture
- Authentication Header
- Encapsulating Security Payload
- Combinations of Security Associations
- Key Management

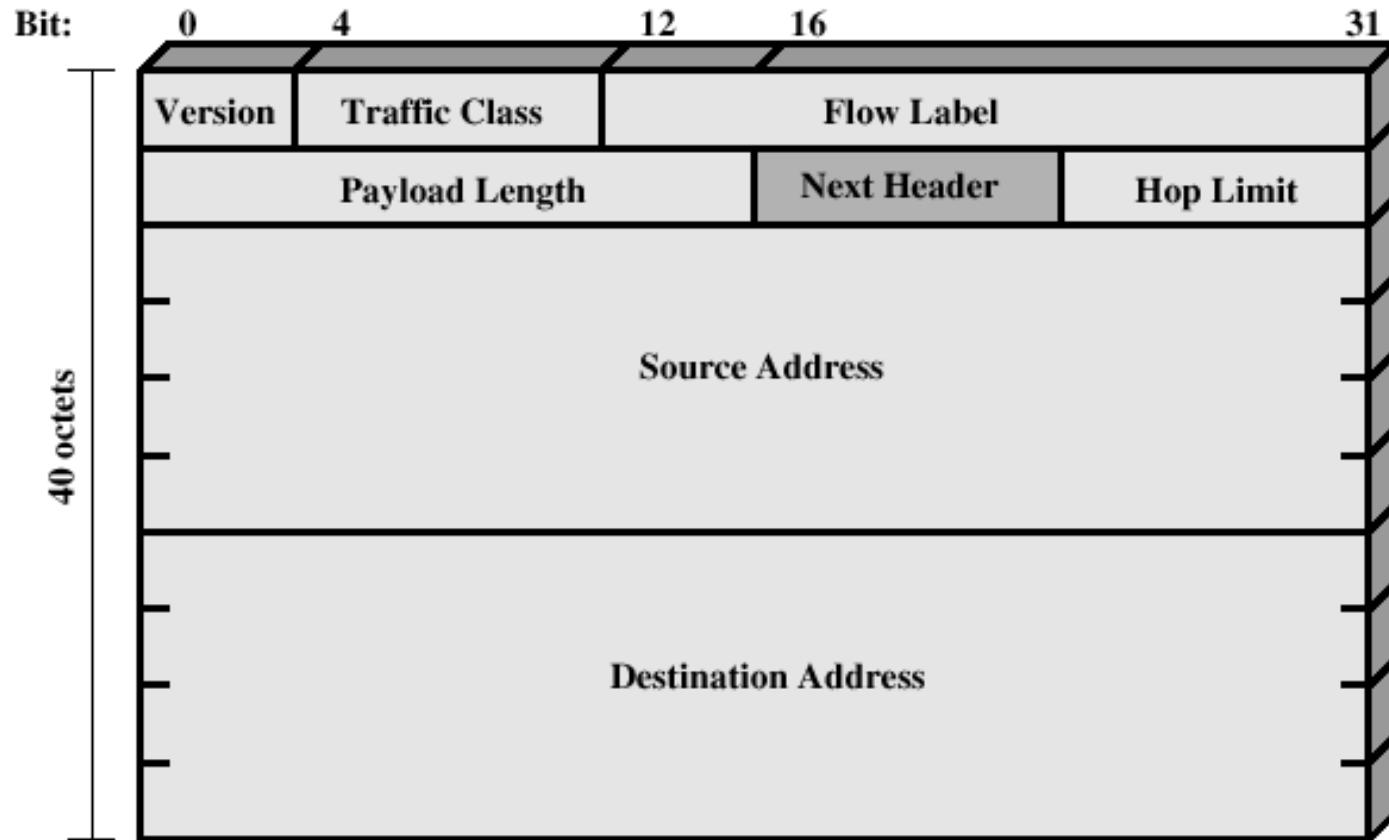
TCP/IP Example



IPv4 Header



IPv6 Header



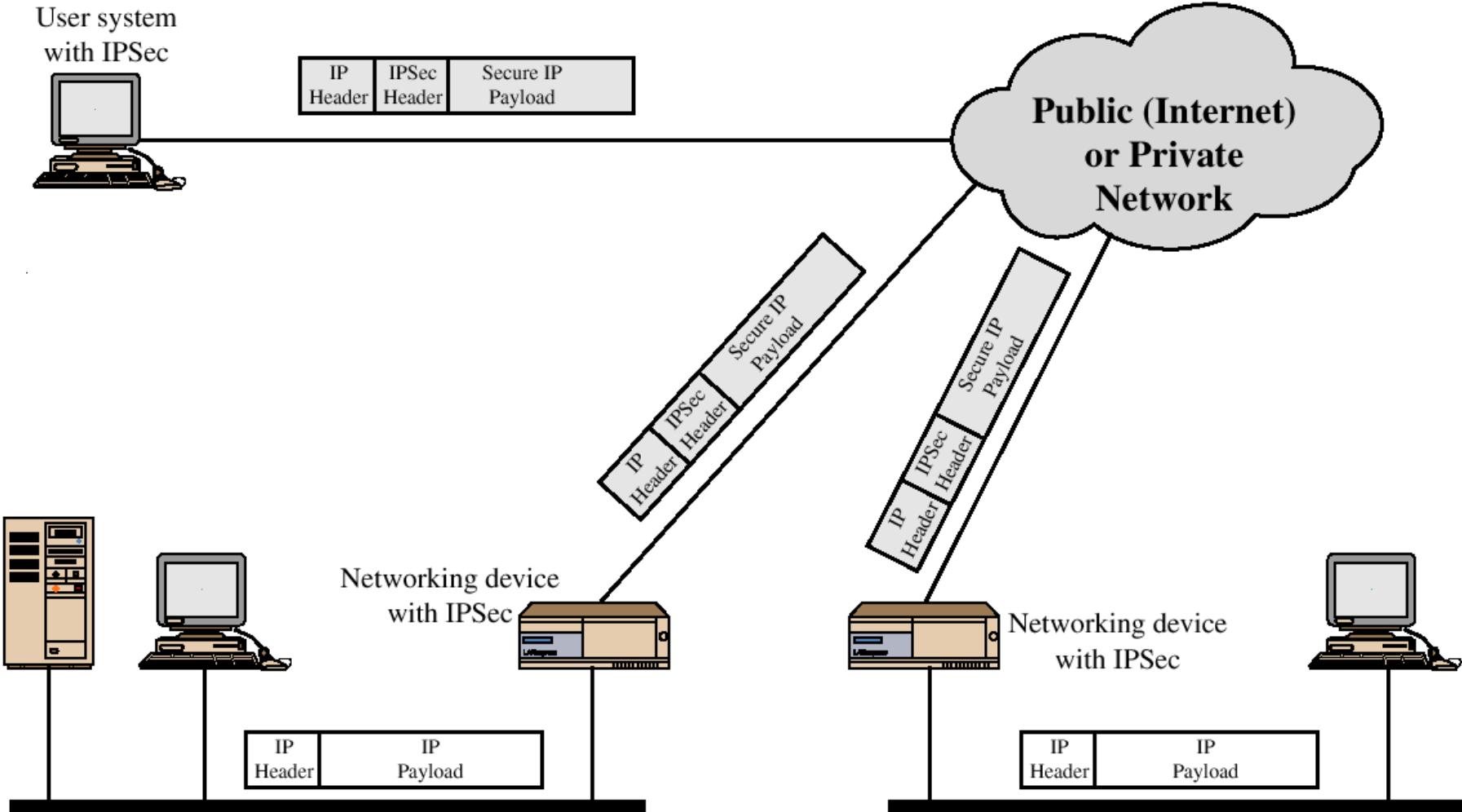
IP Security Overview

IPSec is not a single protocol. Instead, IPSec provides a set of security algorithms plus a general framework that allows a pair of communicating entities to use whichever algorithms provide security appropriate for the communication.

IP Security Overview

- Applications of IPsec
 - Secure branch office connectivity over the Internet
 - Secure remote access over the Internet
 - Establishing extranet and intranet connectivity with partners
 - Enhancing electronic commerce security

IP Security Scenario



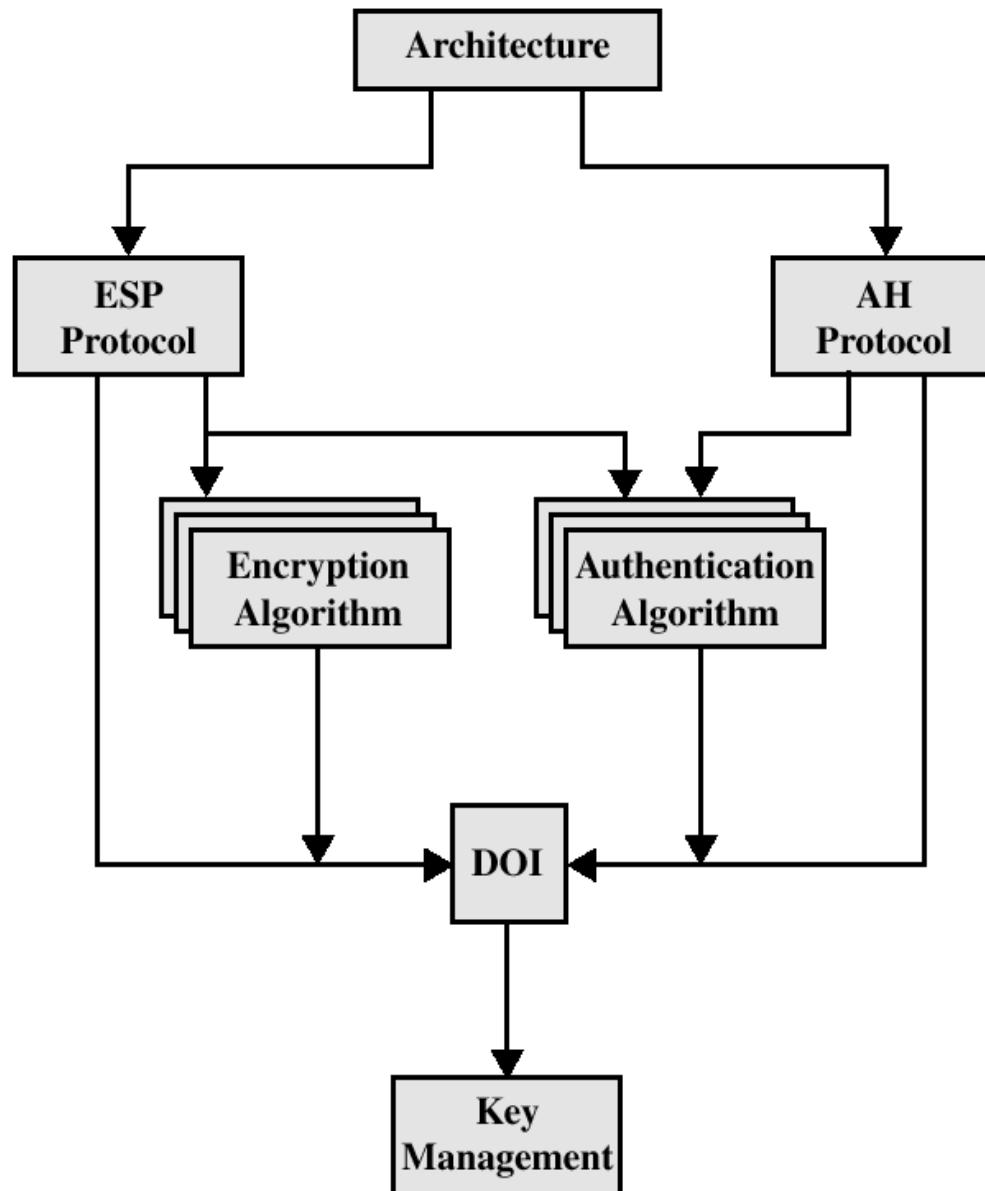
IP Security Overview

- Benefits of IPSec
 - Transparent to applications (below transport layer (TCP, UDP))
 - Provide security for individual users
- IPSec can assure that:
 - A router or neighbor advertisement comes from an authorized router
 - A redirect message comes from the router to which the initial packet was sent
 - A routing update is not forged

IP Security Architecture

- IPSec documents:
 - RFC 2401: An overview of security architecture
 - RFC 2402: Description of a packet encryption extension to IPv4 and IPv6
 - RFC 2406: Description of authentication extension to IPv4 and IPv6
 - RFC 2408: Specification of key management capabilities

IPSec Document Overview



IPSec Services

- Access Control
- Connectionless integrity
- Data origin authentication
- Rejection of replayed packets
- Confidentiality (encryption)
- Limited traffic flow confidentiality

IPSec Services

	AH	ESP (encryption only)	ESP (encryption plus authentication)
Access control	✓	✓	✓
Connectionless integrity	✓		✓
Data origin authentication	✓		✓
Rejection of replayed packets	✓	✓	✓
Confidentiality		✓	✓
Limited traffic flow confidentiality		✓	✓

Security Associations (SA)

- A one way relationship between a sender and a receiver.
- Identified by three parameters:
 - Security Parameter Index (SPI)
 - IP Destination address
 - Security Protocol Identifier

SA Parameters

A security association is normally defined by the following parameters:

- **Sequence Number Counter:** A 32-bit value used to generate the Sequence Number field in AH or ESP headers
- **Sequence Counter Overflow:** A flag indicating whether overflow of the Sequence Number Counter should generate an auditable event and prevent further transmission of packets on this SA
- **Anti-Replay Window:** Used to determine whether an inbound AH or ESP packet is a replay
- **AH Information:** Authentication algorithm, keys, key lifetimes, and related parameters being used with AH

www.rejinpaul.com

SA Parameters

- **ESP Information:** Encryption and authentication algorithm, keys, initialization values, key lifetimes, and related parameters being used with ESP
- **Lifetime of This Security Association:** A time interval or byte count after which an SA must be replaced with a new SA
- **IPSec Protocol Mode:** Tunnel, transport, or wildcard.
- **Path MTU:** Any observed path maximum transmission unit.

Transport Mode

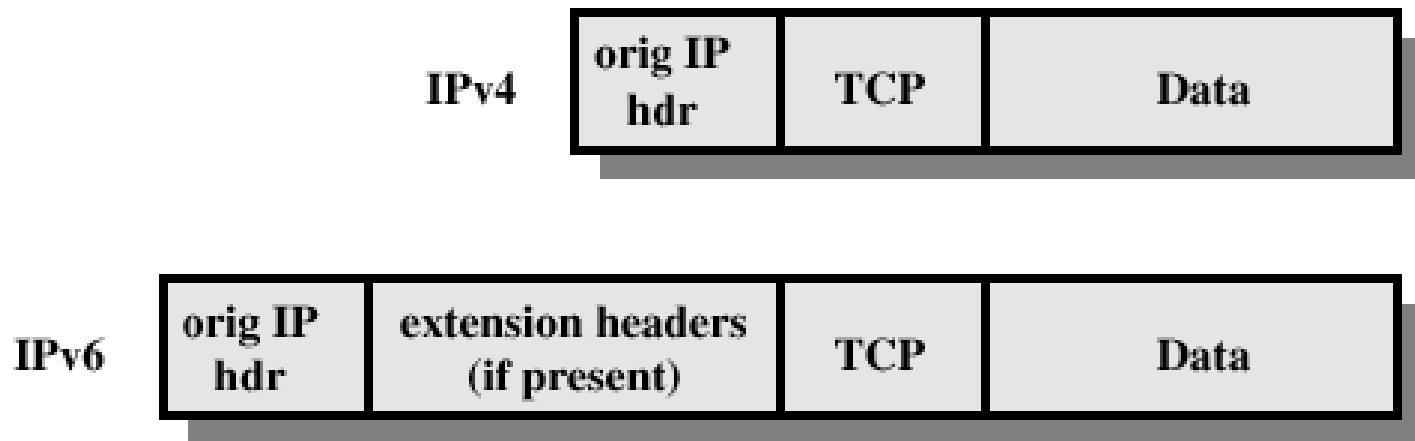
- Transport mode provides protection primarily for upper-layer protocols.
- Examples include a TCP or UDP segment or an ICMP packet, all of which operate directly above IP in a host protocol stack.
 - Used for end-to-end communication between two hosts
- When a host runs AH or ESP over IPv4, the payload is the data that normally follow the IP header.
- For IPv6, the payload is the data that normally follow both the IP header and any IPv6 extensions headers that are present, with the possible exception of the destination options header.
- ESP in transport mode encrypts and optionally authenticates the IP payload but not the IP header.
- AH in transport mode authenticates the IP payload and selected portions of the IP header.

Tunnel Mode

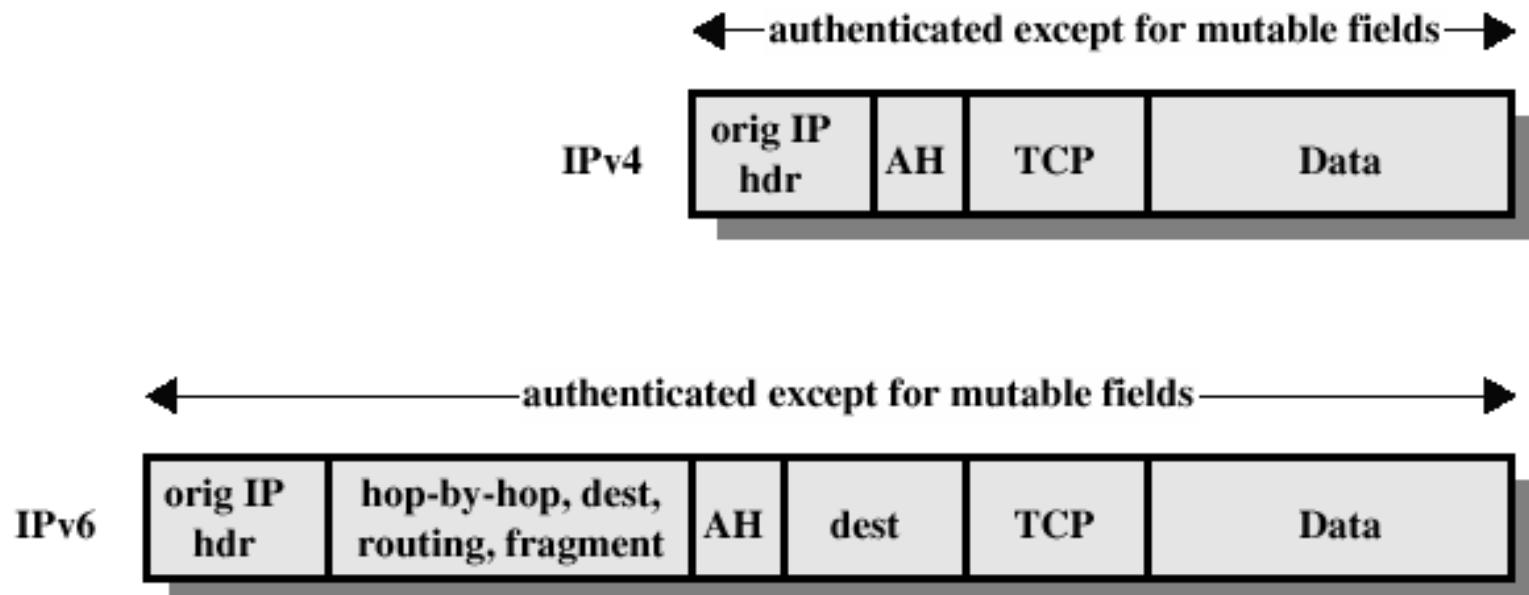
- Tunnel mode provides protection to the entire IP packet.
- To achieve this, after the AH or ESP fields are added to the IP packet, the entire packet plus security fields is treated as the payload of new "outer" IP packet with a new outer IP header.
- The entire original, or inner, packet travels through a "tunnel" from one point of an IP network to another.
- Because the original packet is encapsulated, the new, larger packet may have totally different source and destination addresses, adding to the security.
- Tunnel mode is used when one or both ends of an SA are a security gateway, such as a firewall or router that implements IPSec.

	Transport Mode SA	Tunnel Mode SA
AH	Authenticates IP payload and selected portions of IP header and IPv6 extension headers	Authenticates entire inner IP packet plus selected portions of outer IP header
ESP	Encrypts IP payload and any IPv6 extesion header	Encrypts inner IP packet
ESP with authentication	Encrypts IP payload and any IPv6 extesion header. Authenticates IP payload but no IP header	Encrypts inner IP packet. Authenticates inner IP packet.

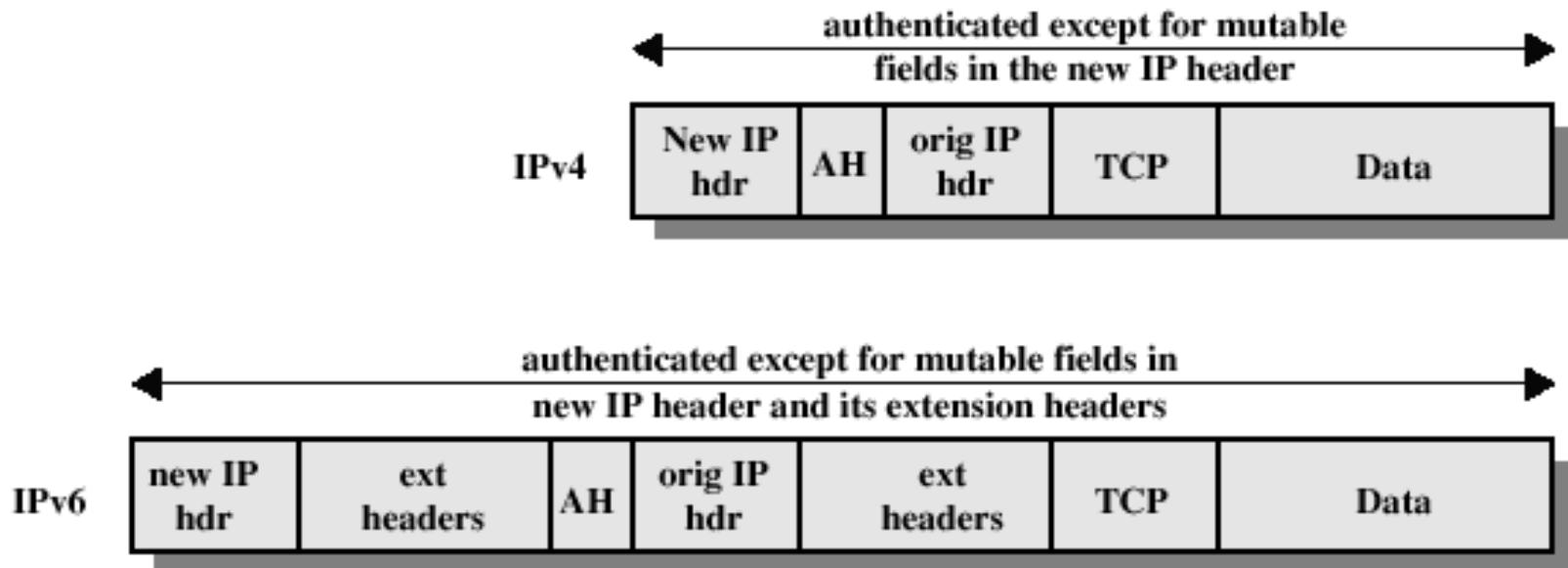
Before applying AH



Transport Mode (AH Authentication)



Tunnel Mode (AH Authentication)



Authentication Header

- Provides support for data integrity and authentication (MAC code) of IP packets.
- Guards against replay attacks.

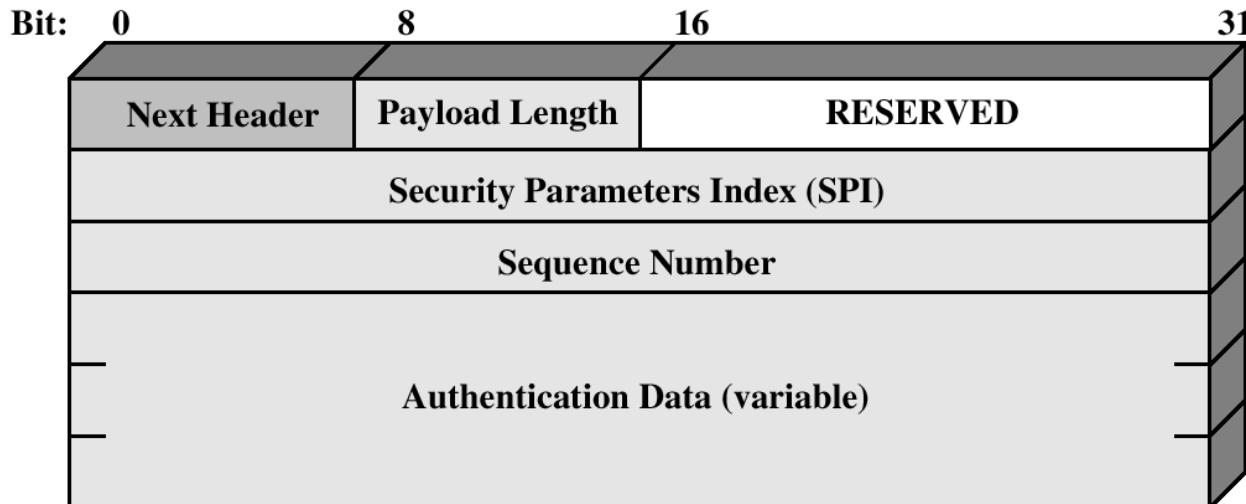
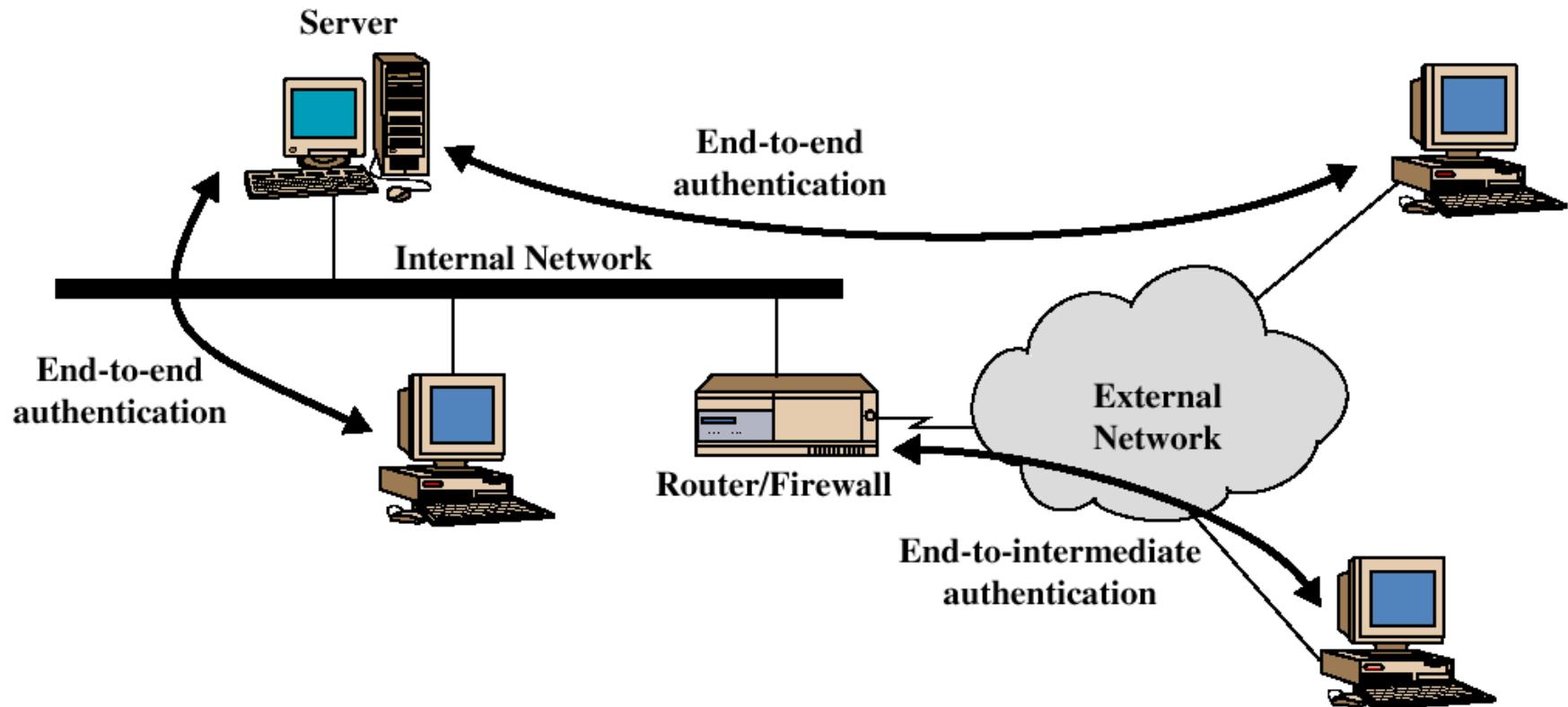


Figure 6.3 IPsec Authentication Header

End-to-end versus End-to-Intermediate Authentication



Encapsulating Security Payload

- ESP provides confidentiality services

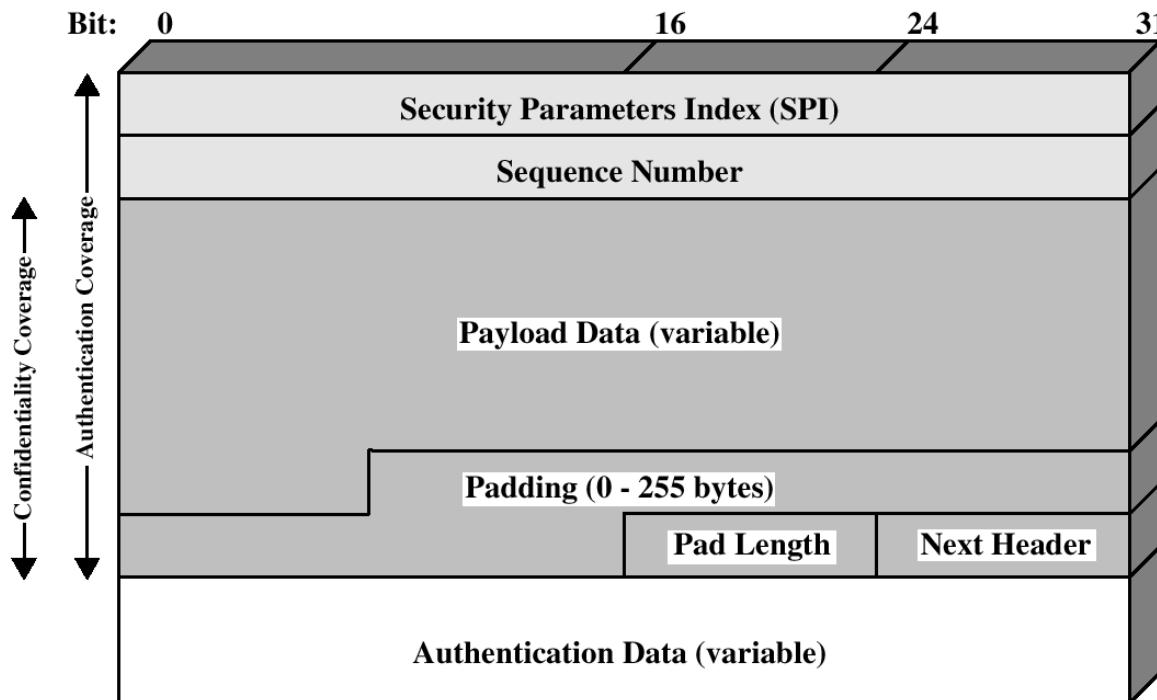
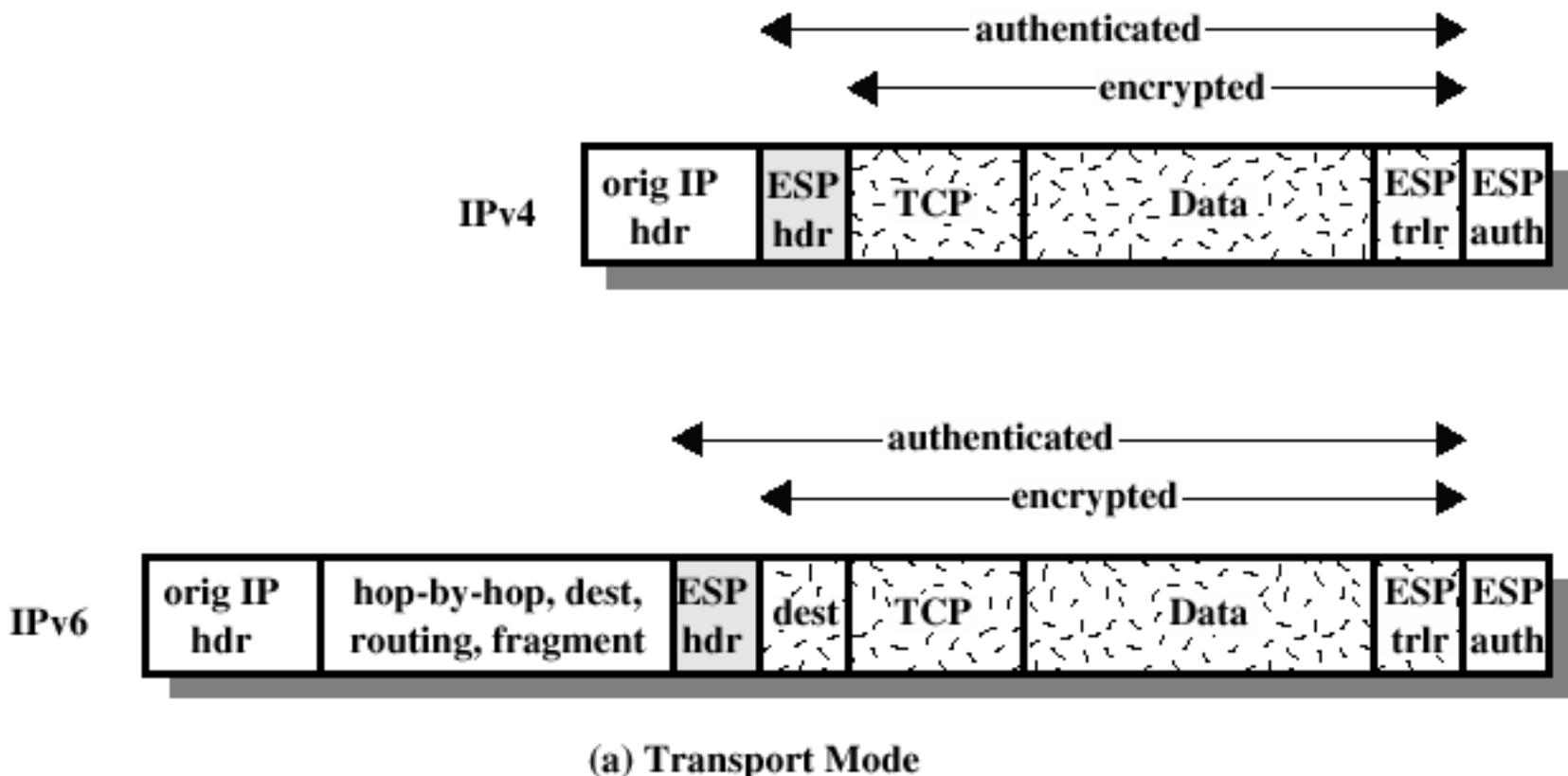


Figure 6.7 IPSec ESP Format

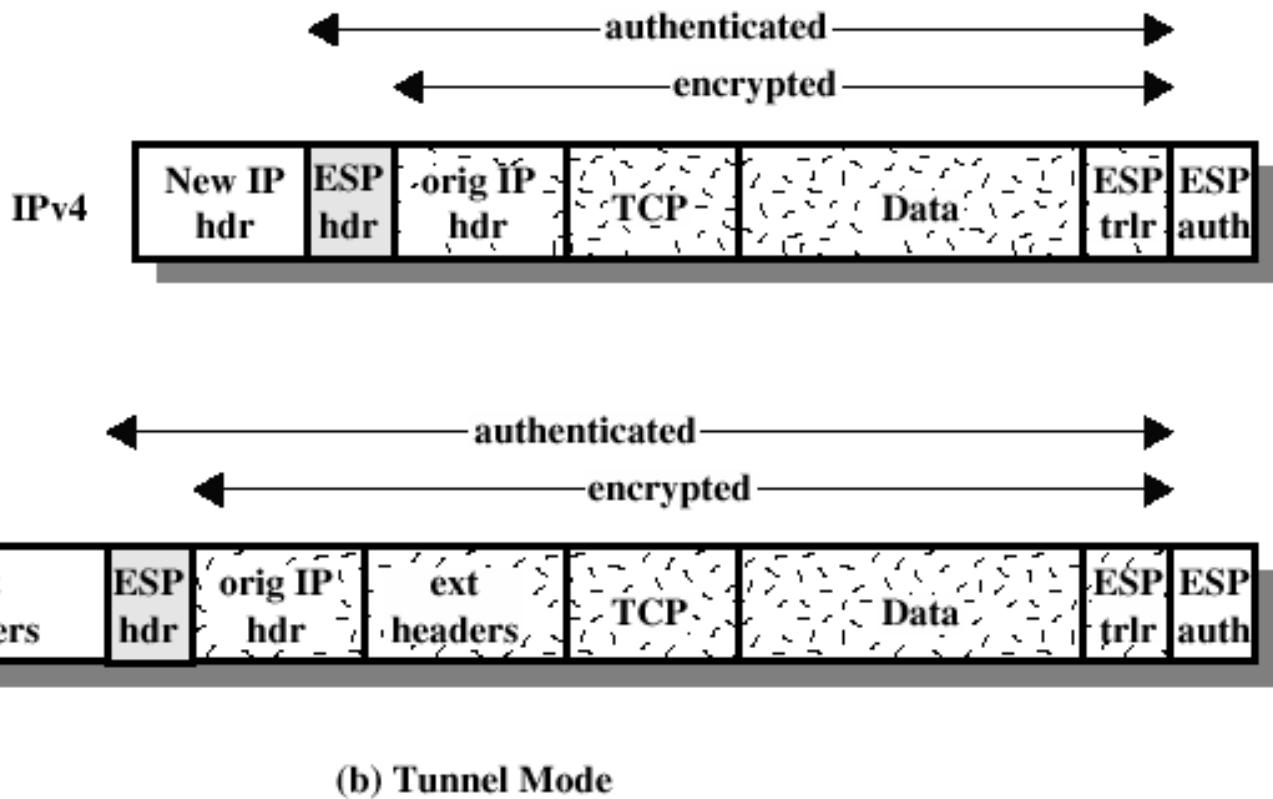
Encryption and Authentication Algorithms

- Encryption:
 - Three-key triple DES
 - RC5
 - IDEA
 - Three-key triple IDEA
 - CAST
 - Blowfish
- Authentication:
 - HMAC-MD5-96
 - HMAC-SHA-1-96

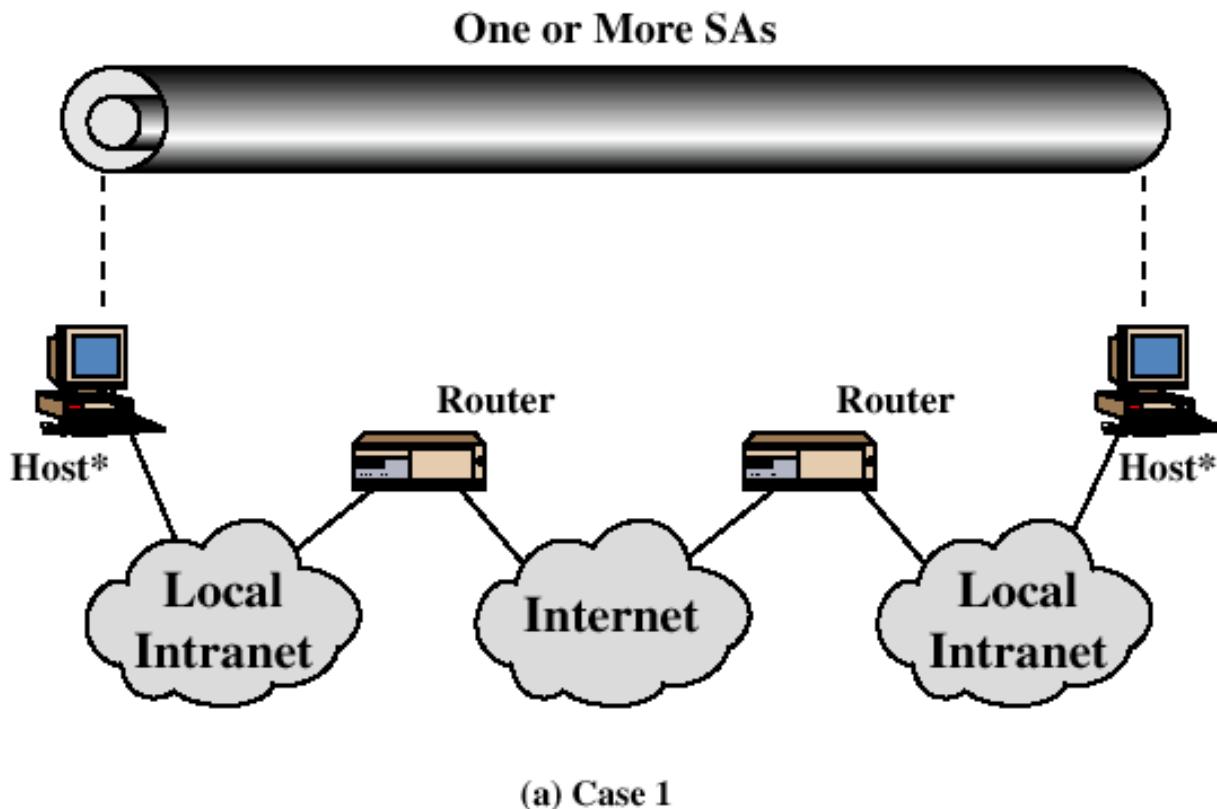
ESP Encryption and Authentication



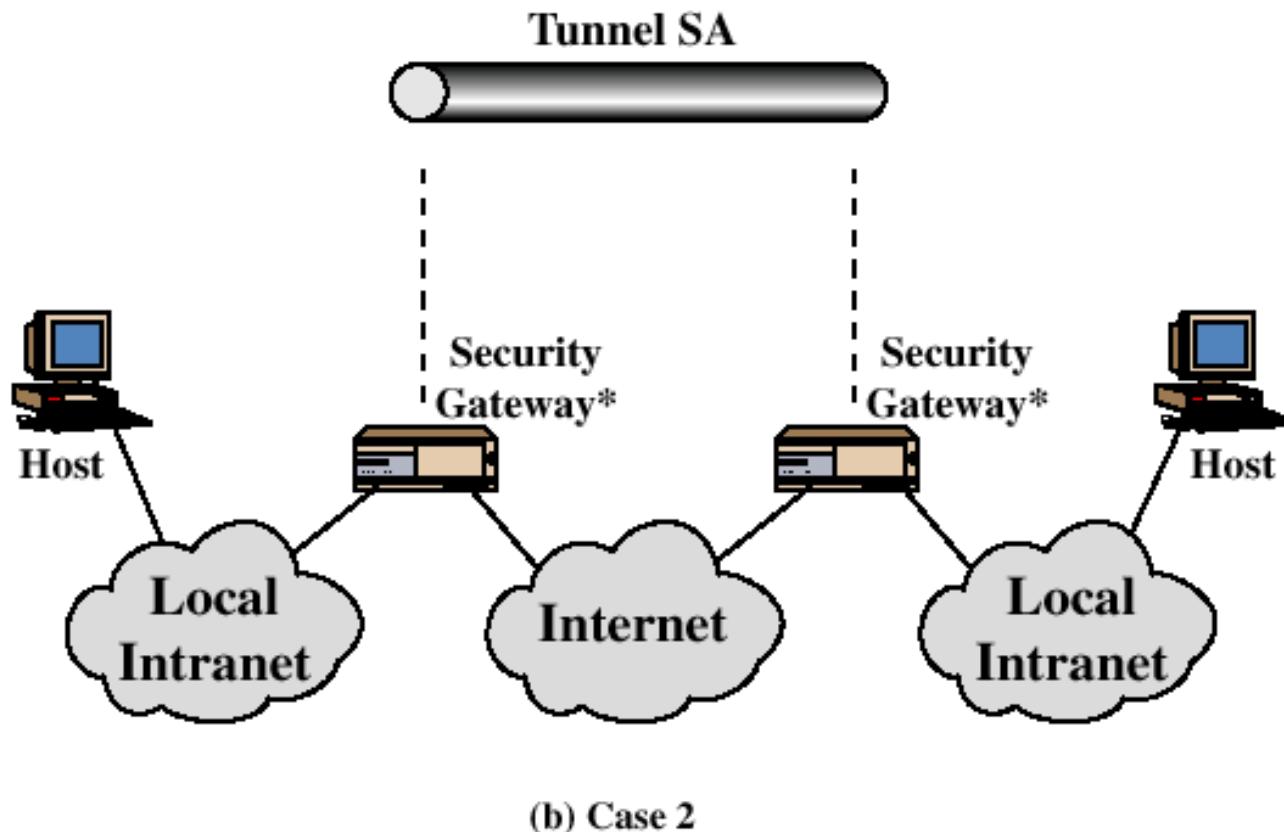
ESP Encryption and Authentication



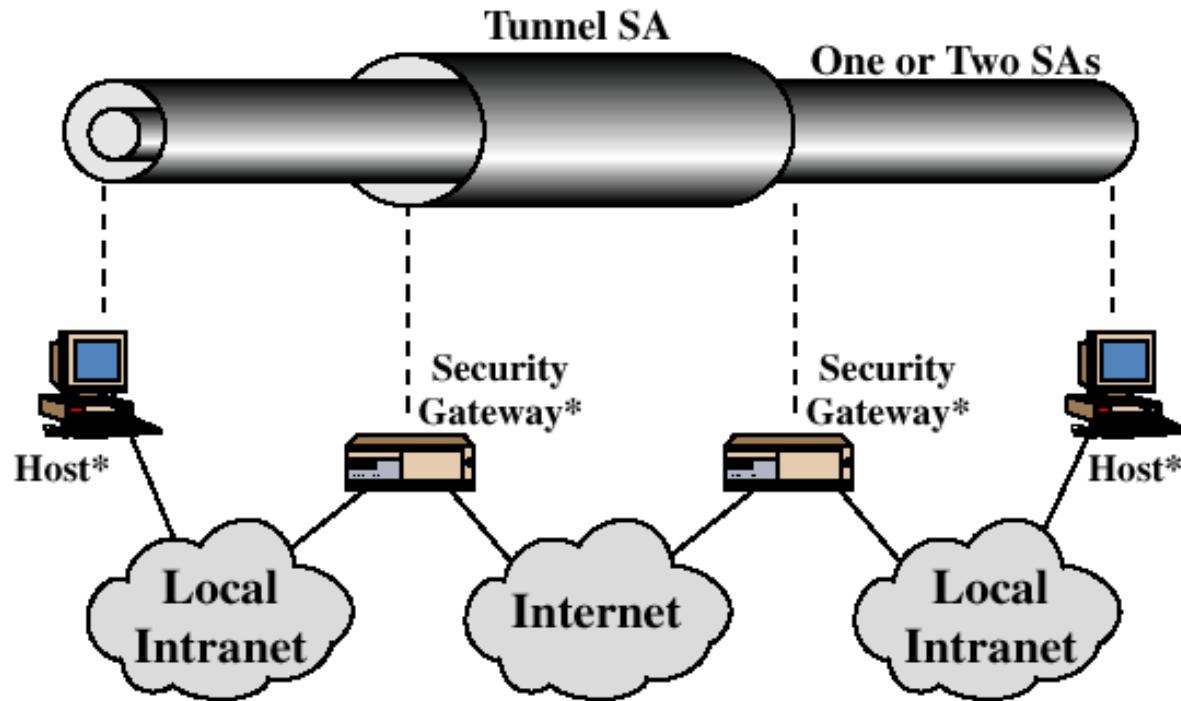
Combinations of Security associations



Combinations of Security Associations

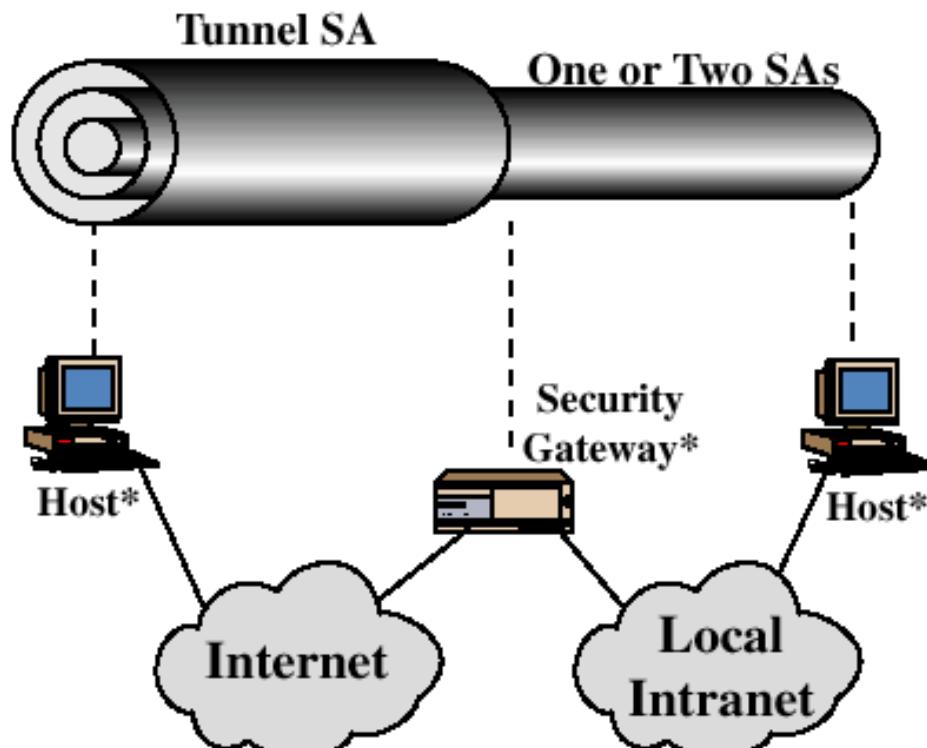


Combinations of Security Associations



(c) Case 3

Combinations of Security Associations



(d) Case 4

Key Management

- Two types:
 - Manual
 - Automated
 - Oakley Key Determination Protocol
 - Internet Security Association and Key Management Protocol (ISAKMP)

WEB Security

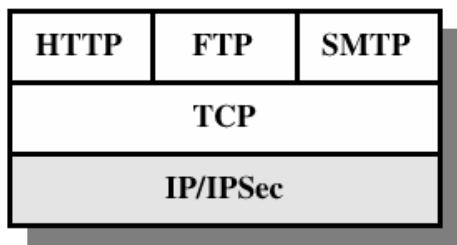
Outline

- Web Security Considerations
- Secure Socket Layer (SSL) and Transport Layer Security (TLS)
- Secure Electronic Transaction (SET)

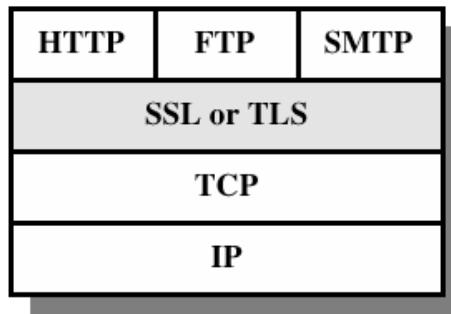
Web Security Considerations

- The WEB is very visible.
- Complex software hide many security flaws.
- Web servers are easy to configure and manage.
- Users are not aware of the risks.

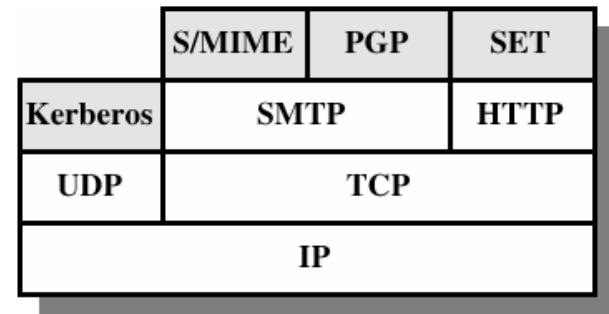
Security facilities in the TCP/IP protocol stack



(a) Network Level



(b) Transport Level



(c) Application Level

SSL and TLS

- SSL was originated by Netscape
- TLS working group was formed within IETF
- First version of TLS can be viewed as an SSLv3.1

SSL Architecture

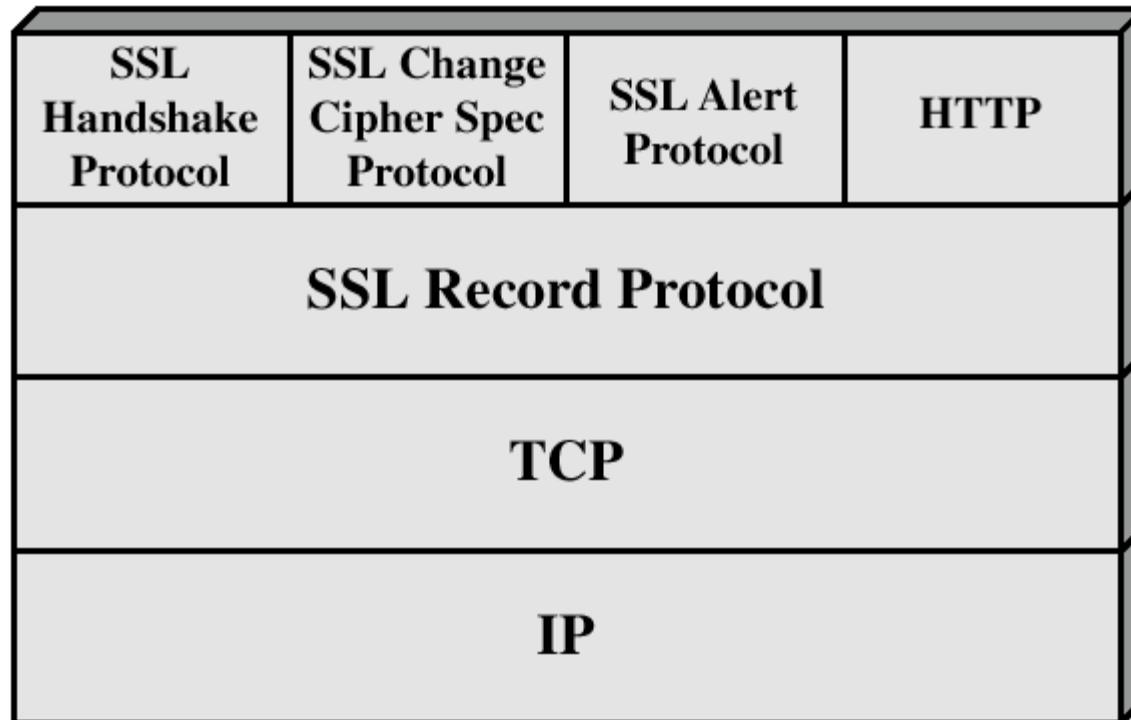
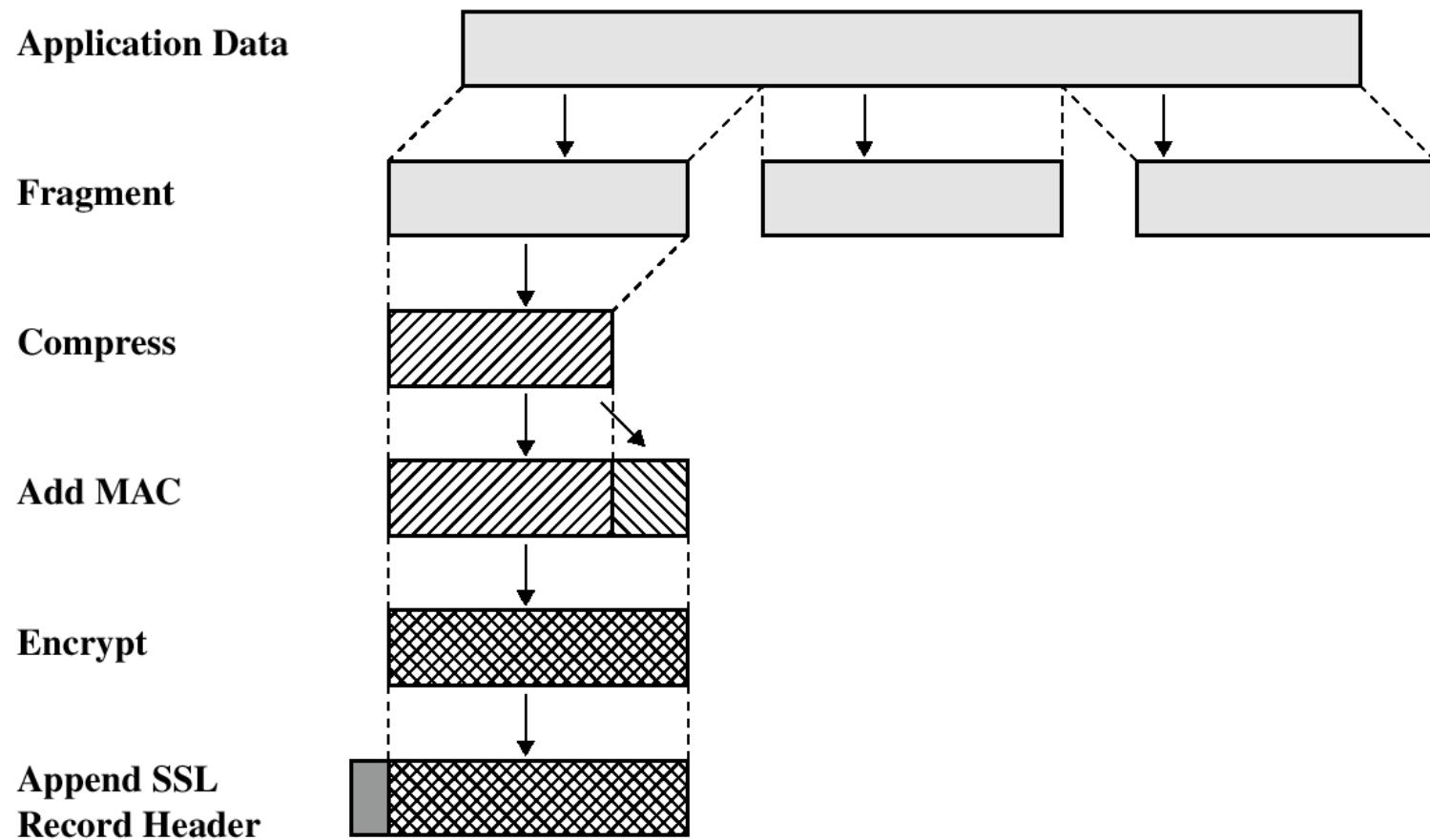
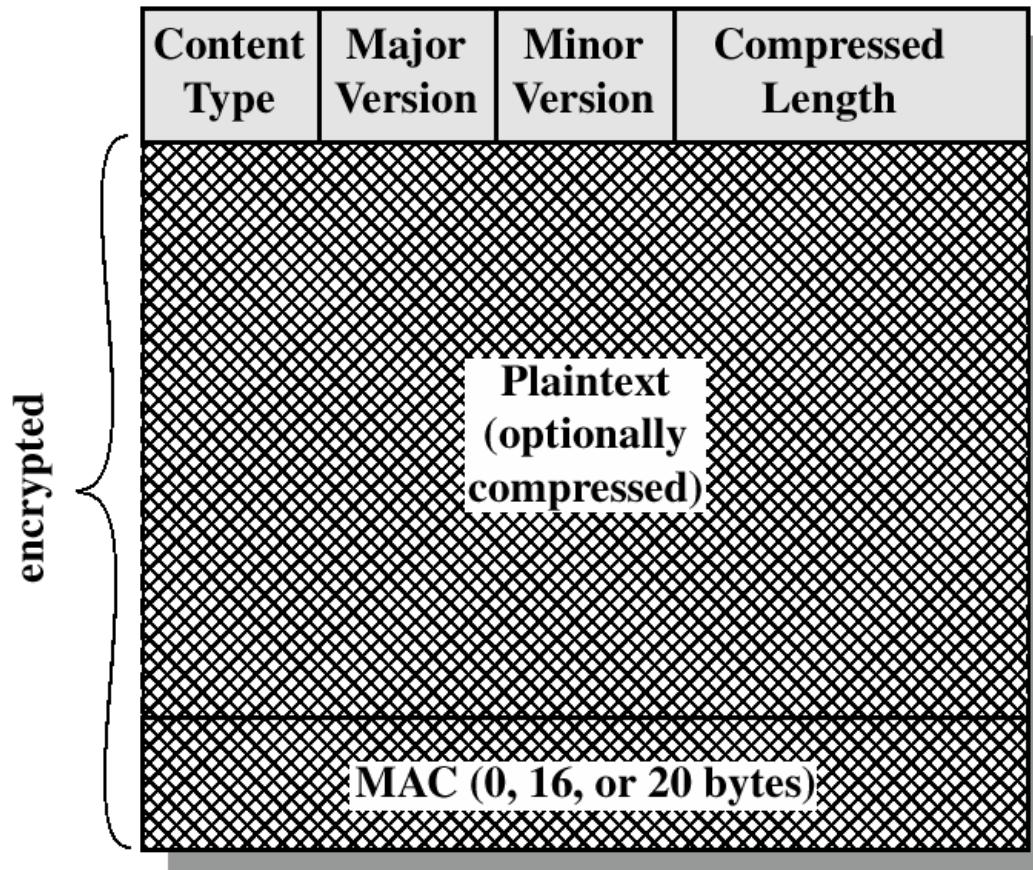


Figure 7.2 SSL Protocol Stack

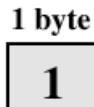
SSL Record Protocol Operation



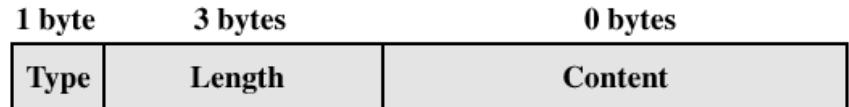
SSL Record Format



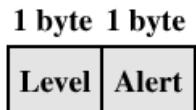
SSL Record Protocol Payload



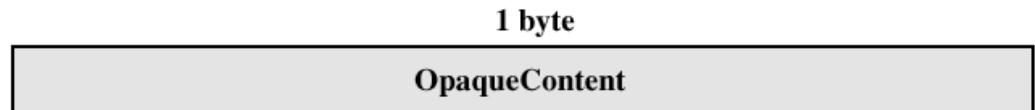
(a) Change Cipher Spec Protocol



(c) Handshake Protocol



(b) Alert Protocol

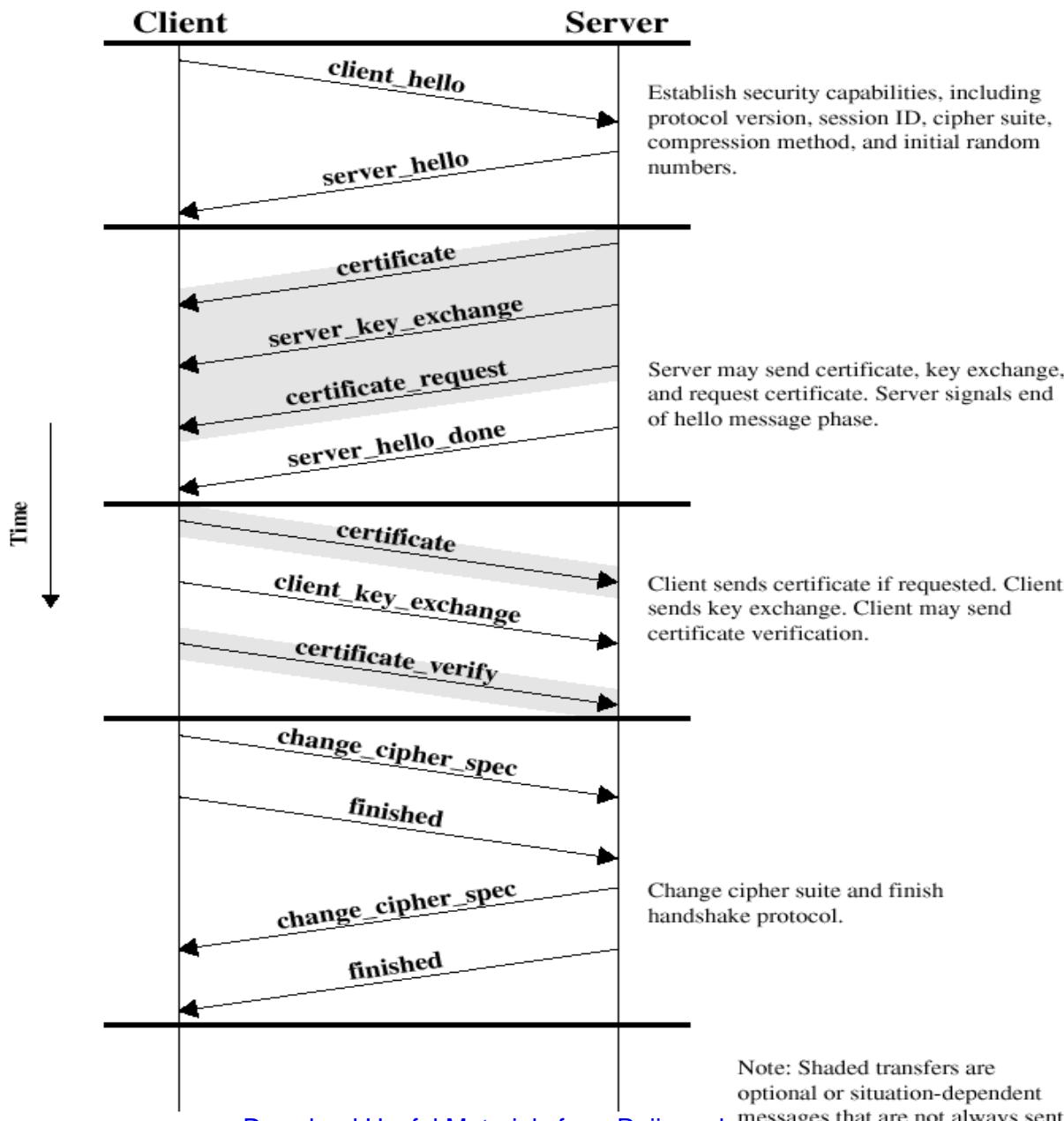


(d) Other Upper-Layer Protocol (e.g., HTTP)

Handshake Protocol

- The most complex part of SSL.
- Allows the server and client to authenticate each other.
- Negotiate encryption, MAC algorithm and cryptographic keys.
- Used before any application data are transmitted.

Handshake Protocol Action



Message Types

Code	Description
0	HelloRequest
1	ClientHello
2	ServerHello
11	Certificate
12	ServerKeyExchange
13	CertificateRequest
14	ServerHelloDone
15	CertificateVerify
16	ClientKeyExchange
20	Finished

Alert level types

Code	Level type	Connection state
1	warning	connection or security may be unstable.
2	fatal	connection or security may be compromised, or an unrecoverable error has occurred.

Transport Layer Security

- The same record format as the SSL record format.
- Defined in RFC 2246.
- Similar to SSLv3.
- Differences in the:
 - version number
 - message authentication code
 - pseudorandom function
 - alert codes
 - cipher suites
 - client certificate types
 - certificate_verify and finished message
 - cryptographic computations
 - padding