CRYPTOGRAPHY AND NETWORK SECURITY

Ref from: Network Security Essentials

by

William Stallings

About This Course

Textbook:

1. Network Security Essentials: Applications and Standards, 3rd Ed. William Stallings
2. Cryptography and Network Security: Principles and Practices, 4th Ed. William Stallings

Contents:

1. Cryptography
   - Algorithms and protocols
     - Conventional and public key-based encryption, hash func, digital signatures, and key exchange
2. Network security applications
   - Applications and tools
     - Kerberos, X.509v3 certificates, PGP, S/MIME, IP security, SSL/TLS, SET, and SNMPv3
3. System security
   - System-level issues
     - Intruders, viruses, worms, DOS

COURSE EDUCATIONAL OBJECTIVES (CEOs)

The main goal of this course is to provide you with a background, foundation, and insight into the many dimensions of information security. This knowledge will serve as basis for further deeper study into selected areas of the field, or as an important component in your further studies and involvement in computing as a whole.

The primary objectives of the course are to help you:

- Understand information security's importance in our increasingly computer-driven world.
- Master the key concepts of information security and how they "work."
- Develop a "security mindset:" learn how to critically analyze situations of computer and network usage from a security perspective, identifying the salient issues, viewpoints, and trade-offs.
- Clearly and coherently communicate (both verbally and in writing) about complex technical topics.

COURSE OUTCOMES (COs)

After completion of the course, the student will be able to:

- Define the concepts and definition of the information security
- Differentiate between several types of Security attacks, Services and Mechanisms
- Identify the threats to information security
- Show how to protect information recourses
- Show how to maintaining and protecting information system

SYLLABUS

- UNIT-I : Introduction, Symmetric Encryption And Message Authentication, and Message Authentication
- UNIT-II: Public Key Cryptography, and Authentication Applications
- UNIT-III: Email privacy and IP Security
- UNIT-IV : Web Security
- UNIT-V : Intruders, Malicious Software, and Firewalls

Tentative Dates for CRT Classes: 18/06/2018 to 30/06/2018

MID-I Examination: 13/08/2018 to 18/08/2018

UNIT-I

PART-I

- Background
- Security Attacks
- Security services
- Security mechanisms
- A model for Internetwork Security
- Internet standards and RFCs

BACKGROUND

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

To give you a feel for the areas covered in this book, consider the following examples of security violations:

- User A transmits a file to user B. The file contains sensitive information (e.g., payroll records) that is to be protected from disclosure. User C, who is not authorized to read the file, is able to monitor the transmission and capture a copy of the file during its transmission.

- A network manager, D, transmits a message to a computer, E, under its management.The message instructs computer E to update an authorization file to include the identities of a number of new users who are to be given access to that computer. User F intercepts the message, alters its contents to add or delete entries, and then forwards the message to E, which accepts the message as coming from manager D and updates its authorization file accordingly.

- Rather than intercept a message, user F constructs its own message with the desired entries and transmits that message to E as if it had come from manager D. Computer E accepts the message as coming from manager D and updates its authorization file accordingly.

- An employee is fired without warning. The personnel manager sends a message to a server system to invalidate the employee's account. When the invalidation is accomplished, the server is to post a notice to the employee's file as confirmation of the action. The employee is able to intercept the message and delay it long enough to make a final access to the server to retrieve sensitive information. The message is then forwarded, the action taken, and the confirmation posted. The employee's action may go unnoticed for some considerable time.

- A message is sent from a customer to a stockbroker with instructions for various transactions. Subsequently, the investments lose value and the customer denies sending the message.

3 Aspects of Info Security

- **Security Attack-** Any action that compromises the security of information owned by an organization
- **Security Mechanism-** A mechanism that is designed to detect, prevent, or recover from a security attack.
- **Security Service-** A service that enhances the security of data processing systems and information transfers of an organization.

- Services are intended to counter security attacks and they make use of one or more security mechanisms to provide the service

SECURITY ATTACKS

Two generic types of attacks:

- A **Passive Attack** attempts to learn or make use of information from the system but does not affect system resources.

- An **Active Attack** attempts to alter system resources or affect their operation

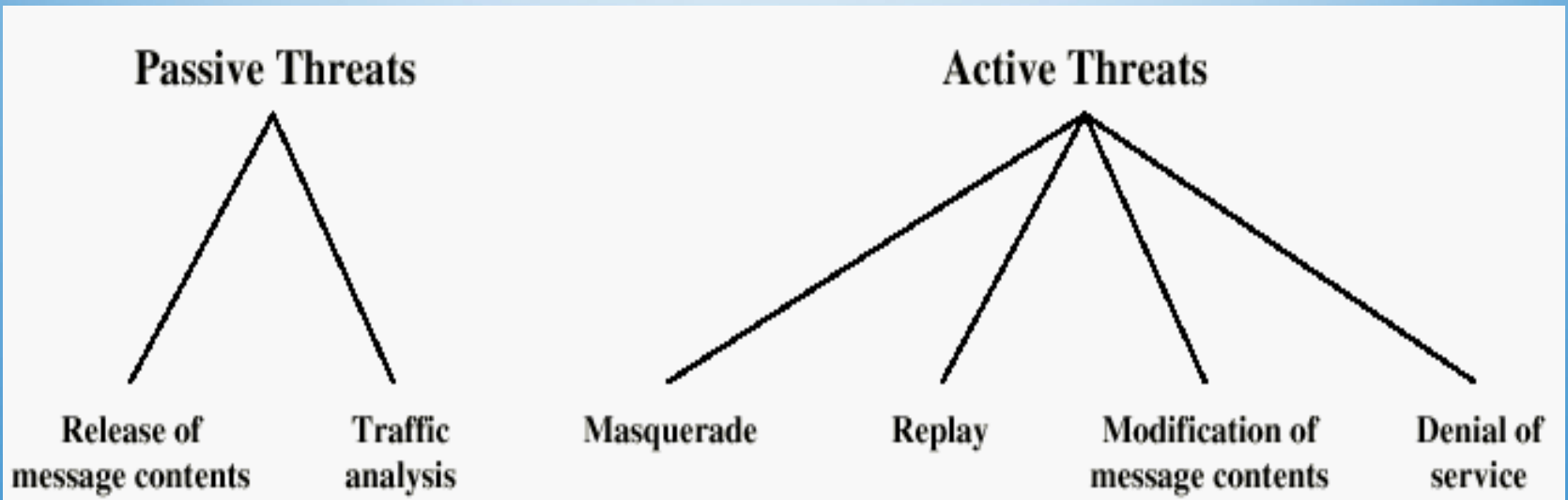
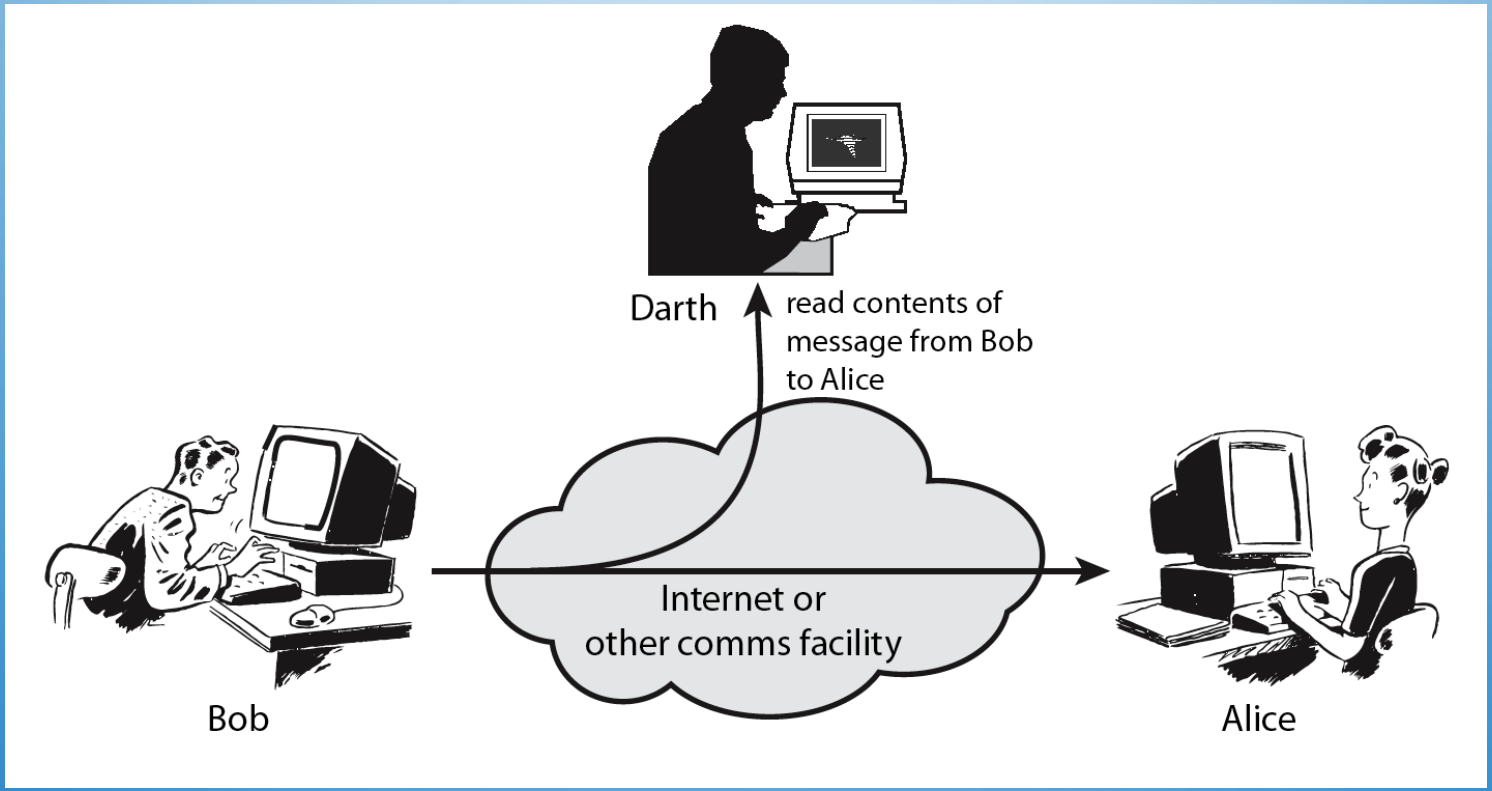


Figure 1.2 Active and Passive Security Threats

SECURITY ATTACKS

Passive – In the nature of eavesdropping on, or monitoring of, transmissions

Goal: Obtain information that is being transmitted



Cont..

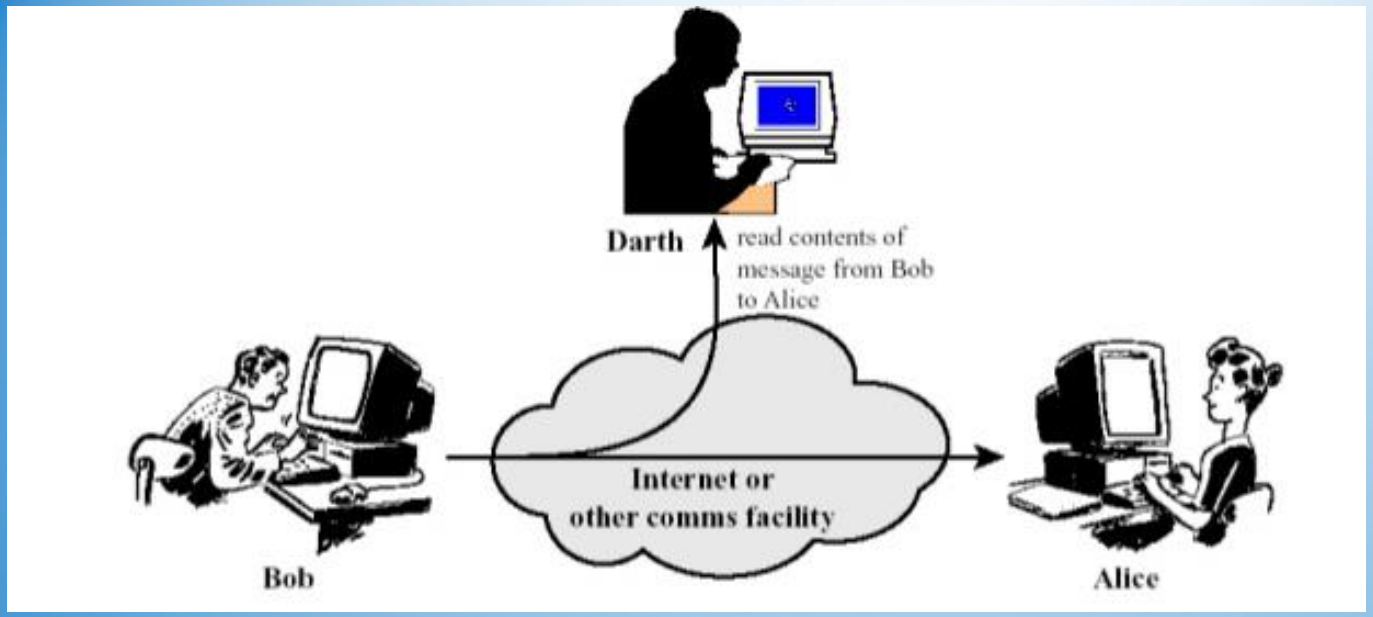
Two types of Passive Attacks:

- Release of Message Contents- Read the content of the message
- Traffic Analysis - Identifying the pattern of messages

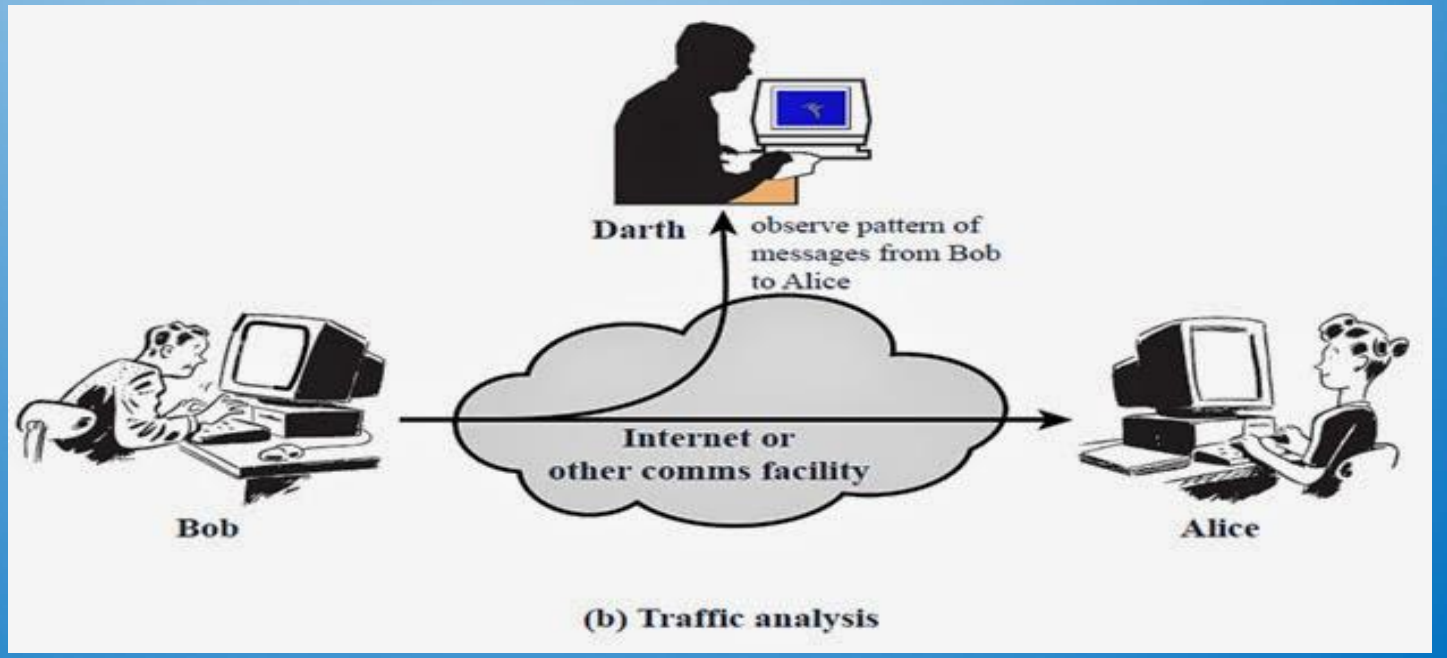
Solution- Mask the content of the message- Encryption

Problem

- If we had encryption Protection in place, an opponent might still be able to 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.

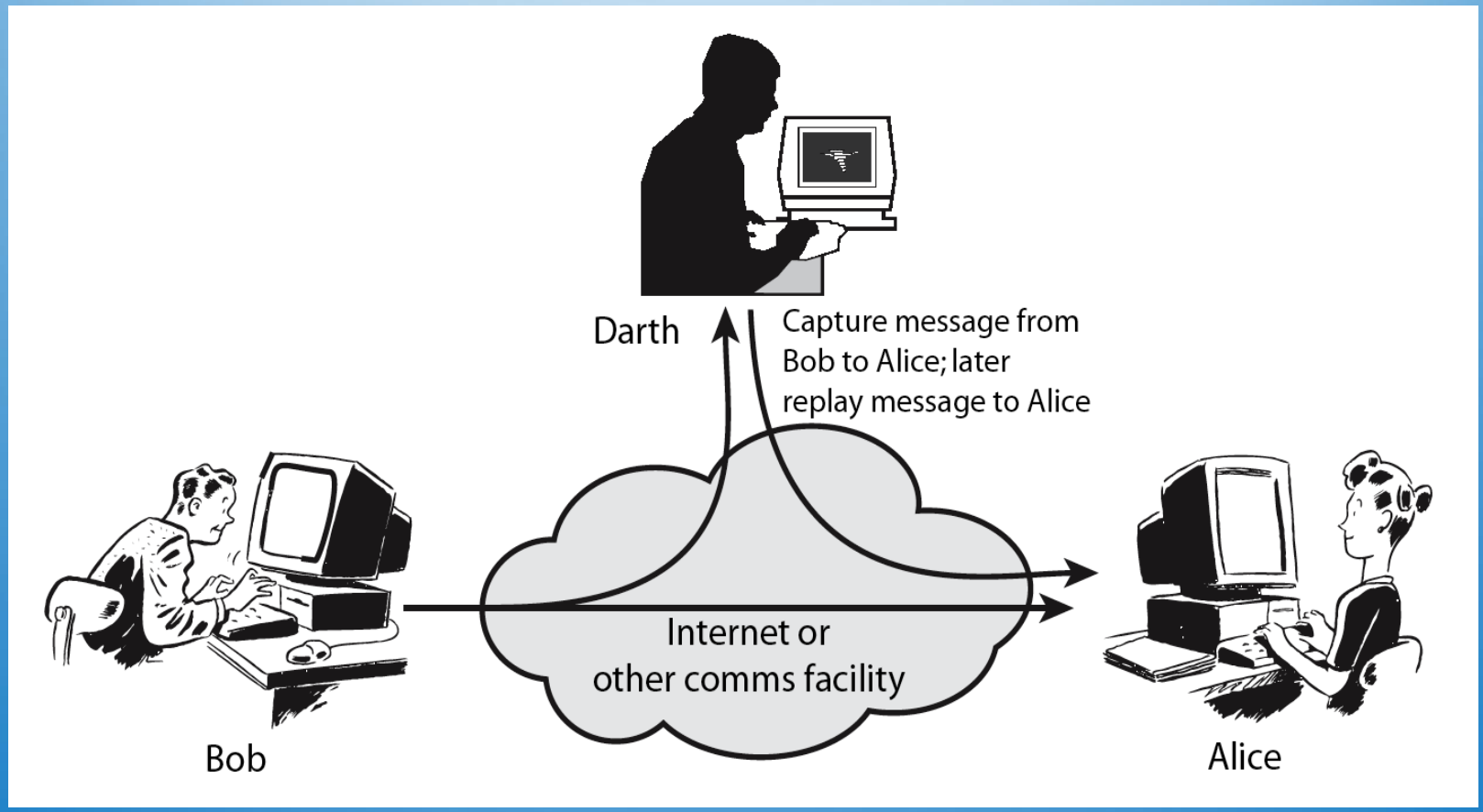


(a)Release of message content



SECURITY ATTACKS

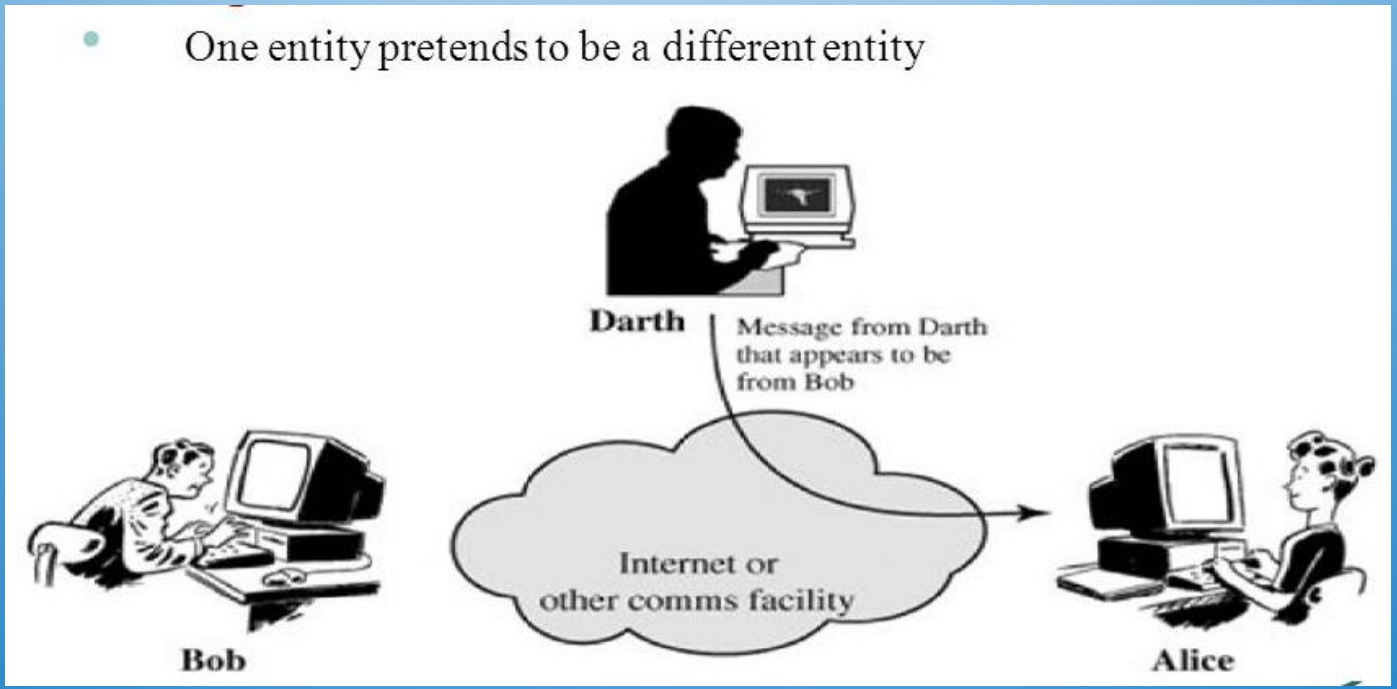
Active – It involves in some modification of the data stream or the creation of a false stream and can be sub divided into four categories



Cont…

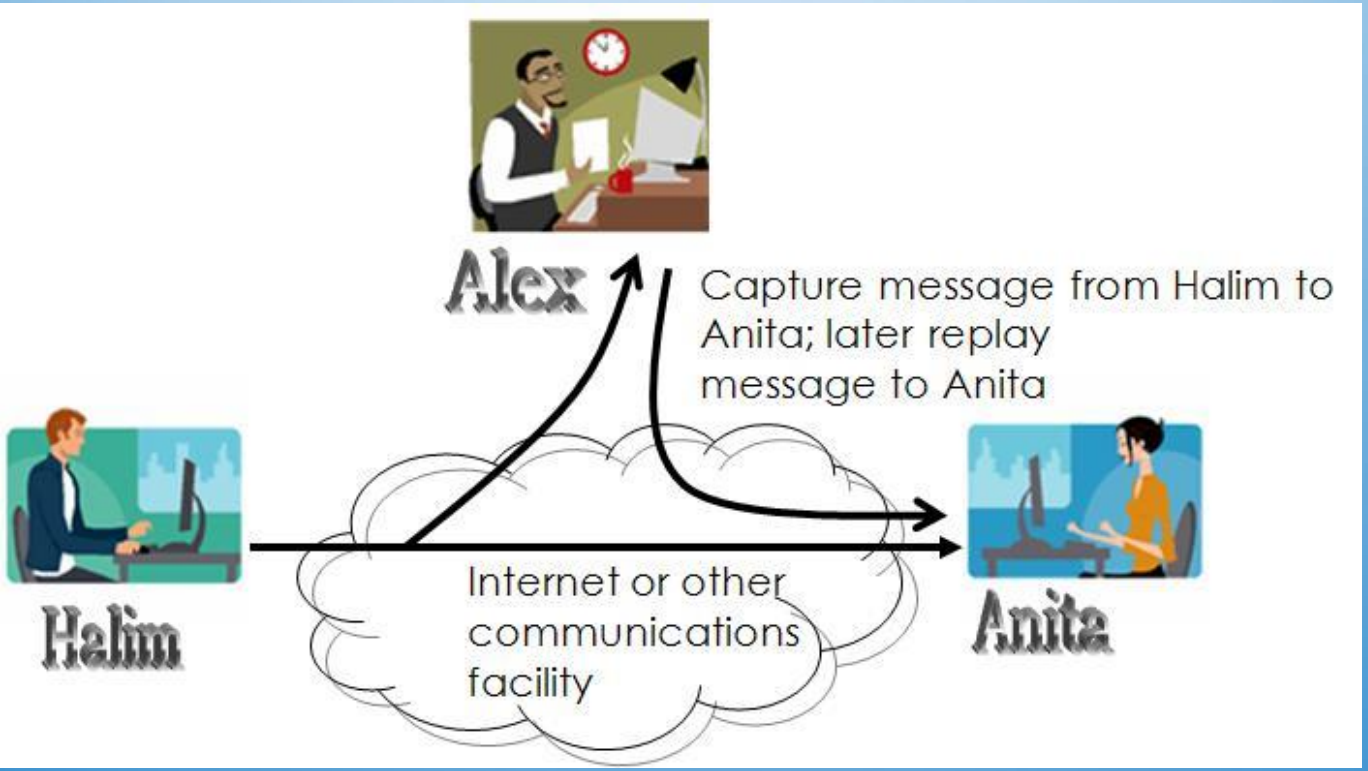
Masquerade –It takes place when one entity pretends to be a different entity

- Authentication Sequence can be captured
- Replayed after a valid authentication sequence has taken place
- Enabling an authorized entity with few privileges to obtain extra privileges by impersonating an entity that has those privileges.



Cont…

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

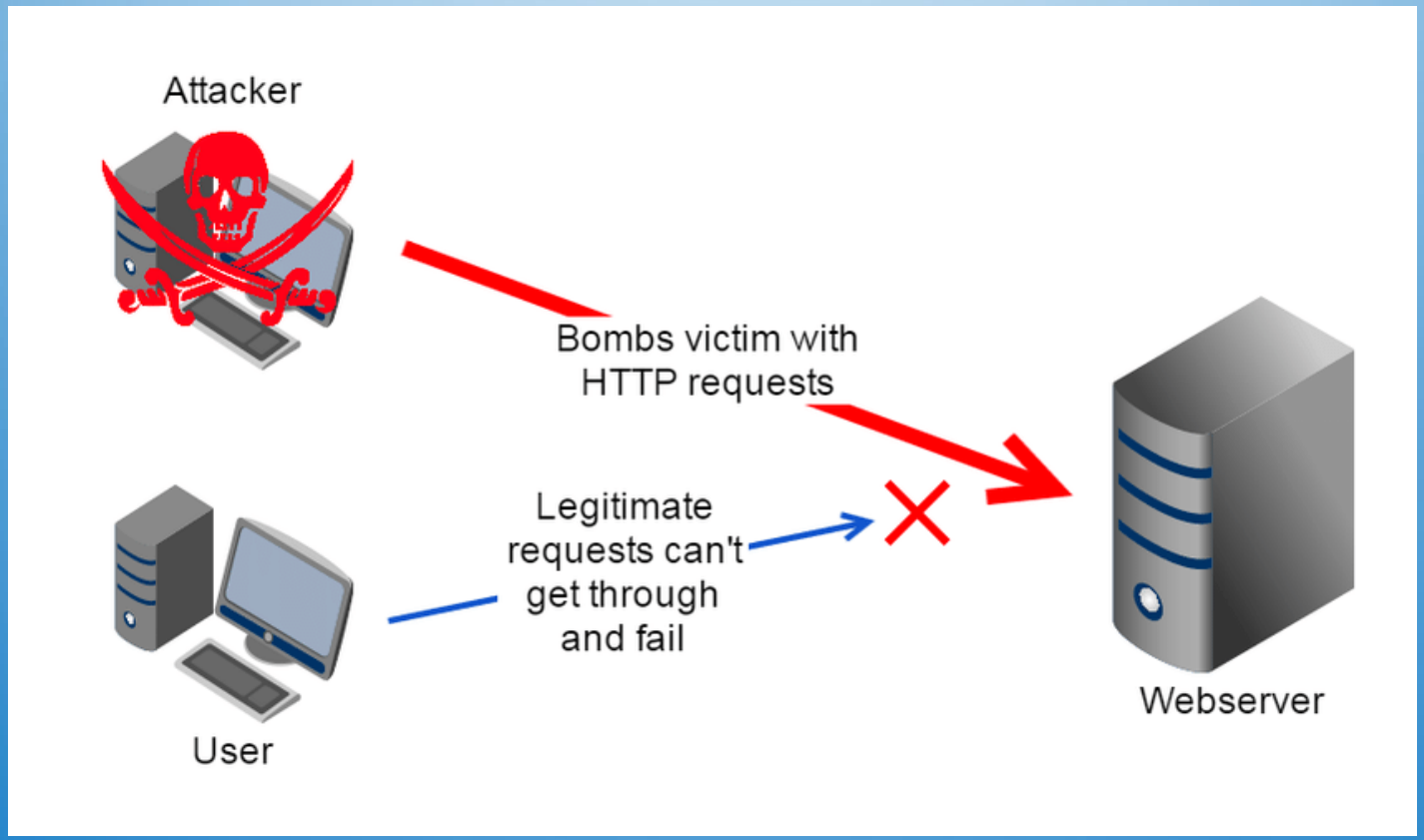


Cont…

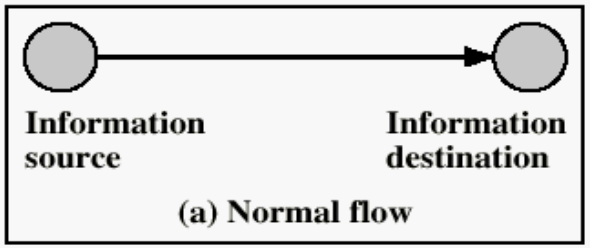
Modification of Message – Some portion of legitimate message is altered, or the messages are delayed or reordered to produce an unauthorized effect

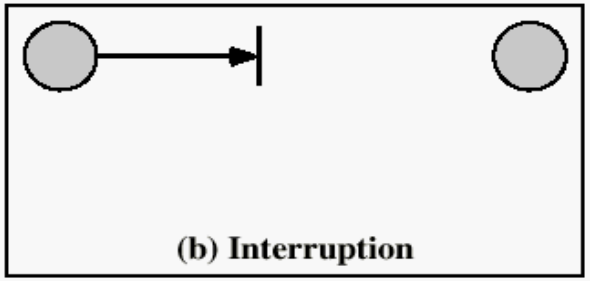
Cont…

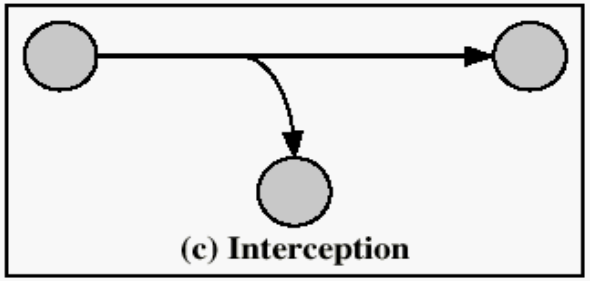
Denial Of Service - It prevents or inhibits the normal use or management of communications facilities

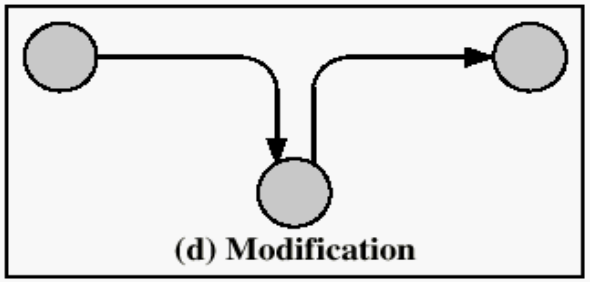


SECURITY ATTACK CLASSIFICATION









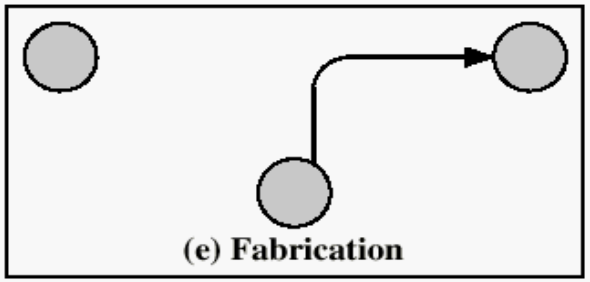
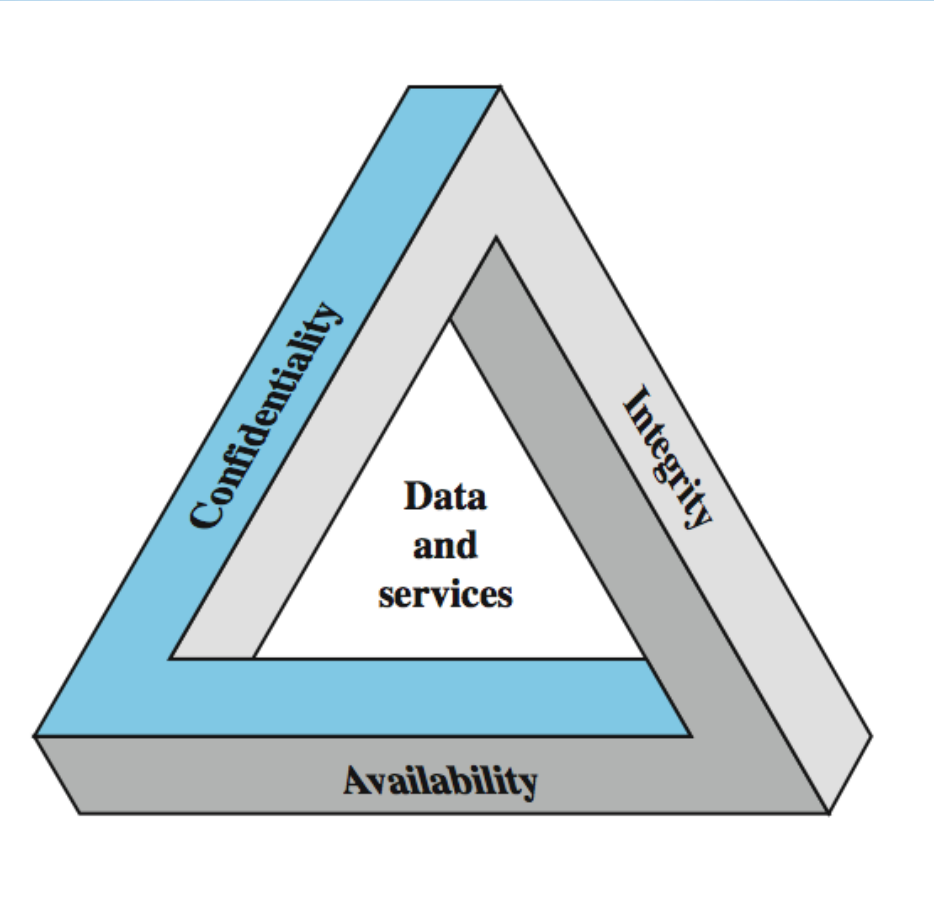


Figure 1.1 Security Threats

SECURITY ATTACKS

- **Interruption:** This is an attack on availability
- **Interception:** This is an attack on confidentiality
- **Modification:** This is an attack on integrity
- **Fabrication:** This is an attack on authenticity

3 PRIMARY SECURITY GOALS



Fundamental security objectives for both data and information/computing services

SECURITY SERVICES

X.800

- A service provided by a protocol layer of communicating open systems, which ensures adequate security of the systems or of data transfers

- Confidentiality (privacy)
- Authentication (who created or sent the data)
- Integrity (has not been altered)
- Non-repudiation (the order is final)
- Access control (prevent misuse of resources)
- Availability (permanence, non-erasure)
  - Denial of Service Attacks
  - Virus that deletes files

AUTHENTICATION- The assurance that the communicating entity is the one that it claims to be.

- Peer Entity Authentication: Used in association with a logical connection to provide confidence in the identity of the entities connected.
- Data Origin Authentication: In a connectionless transfer, provides assurance that the source of received data is as claimed.

ACCESS CONTROL- The prevention of unauthorized use of a resource

DATA CONFIDENTIALITY- The protection of data from unauthorized disclosure

- Connection Confidentiality: The protection of all user data on a connection
- Connectionless confidentiality: The protection of all user data in a single data block
- Selective-Field Confidentiality: The confidentiality of selected fields within the user data on a connection or in a single data block
- Traffic-Flow confidentiality: The protection of the information that might be desired from observation of traffic flows.

DATA INTEGRITY- The assurance that data received are exactly as sent by an authorized entity.

- Connection Integrity with Recovery: Provides for the integrity of all user data on a connection and detects any modification, insertion, deletion, or replay of any data with in an entire data sequence, with recovery attempted
- Connection Integrity without Recovery: As above, but provides only detection without Recovery
- Selective-field Connection Integrity: Provides for the integrity of selected fields within the user data of a data block transferred over a connection and takes the form of determination of whether the selected fields have been modified, inserted, delayed, or replayed.
- Connectionless Integrity: Provides for the integrity of a single connectionless data block and may take the form of detection of data modification. Additionally a limited form of replay detection may be provided.
- Selective-field Connectionless Integrity: Provides for the integrity of selected fields within a single connectionless data block; takes the form of determination of whether the selected fields have been modified

NONREPUDIATION- Provides protection against denial by on of the entities involved in a communication of having participated in all or part or the communication

- Nonrepudiation, Origin: Proof that the message was sent by the specified party
- Nonrepudiation, Destination: Proof of that message was received by the specified party.

AVILABILITY- Both X.800 and RFC 2828 define availability to be the property of a system or a system resource being accessible and usable upon demand by an authorized system entity, according to performance specifications for the system

- A variety of attacks can result in the loss of or reduction in availability.
- Some of these attacks are amenable to automated countermeasures, such as authentication and encryption, whereas others require some sort of physical action to prevent or recover from loss of availability of elements of a distributed system.

Security Mechanism

- Features designed to detect, prevent, or recover from a security attack
- No single mechanism that will support all services required
- One particular element underlies many of the security mechanisms in use:
  - **Cryptographic techniques**
  - Hence we will focus on this topic first

Security Mechanisms (X.800)

- **Specific security mechanisms:** May be incorporated into the appropriate protocol layer in order to provide some of the OSI security services.

- **Pervasive security mechanisms:**Mechanisms that are not specific to any particular OSI security service or protocol layer.

Specific security mechanisms

- **Encipherment**: The use of mathematical algorithms to transform data into a form that is not readily intelligible. The transformation and subsequent recovery of the data depend on an algorithm and zero or more encryption keys.

- **Digital Signature:** Data appended to, or a cryptographic transformation of, a data unit that allows a recipient of the data unit to prove the source and integrity of the data unit and protect against forgery (e.g., by the recipient)

- **Access Control:** A variety of mechanisms that enforce access rights to resources.

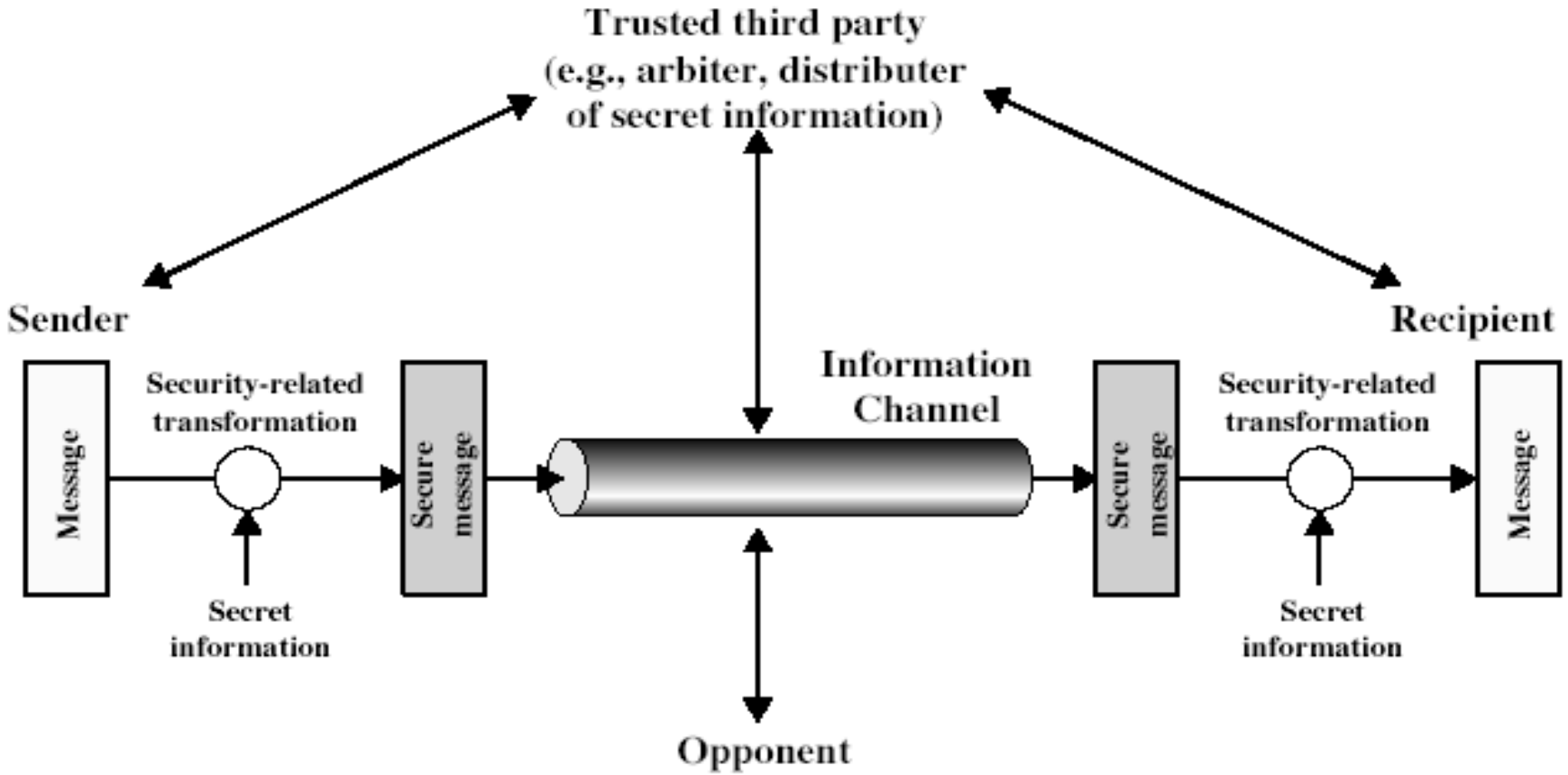
- **Data Integrity:** A variety of mechanisms used to assure the integrity of a data unit or stream of data units.
- **Authentication Exchange:** A mechanism intended to ensure the identity of an entity by means of information exchange.
- **Traffic Padding:** The insertion of bits into gaps in a data stream to frustrate traffic analysis attempts.
- **Routing Control:** Enables selection of particular physically secure routes for certain data and allows routing changes, especially when a breach of security is suspected.

- **Notarization:** The use of a trusted third party to assure certain properties of a data exchange.

Pervasive security mechanisms

- **Trusted Functionality:** That which is perceived to be correct with respect to some criteria (e.g., as established by a security policy).
- **Security Label:** The marking bound to a resource (which may be a data unit) that names or designates the security attributes of that resource.
- **Event Detection:** Detection of security-relevant events.
- **Security Audit Trail:** Data collected and potentially used to facilitate a security audit, which is an independent review and examination of system records and activities.
- **Security Recovery:** Deals with requests from mechanisms, such as event handling and management functions, and takes recovery actions.

Model for Network Security

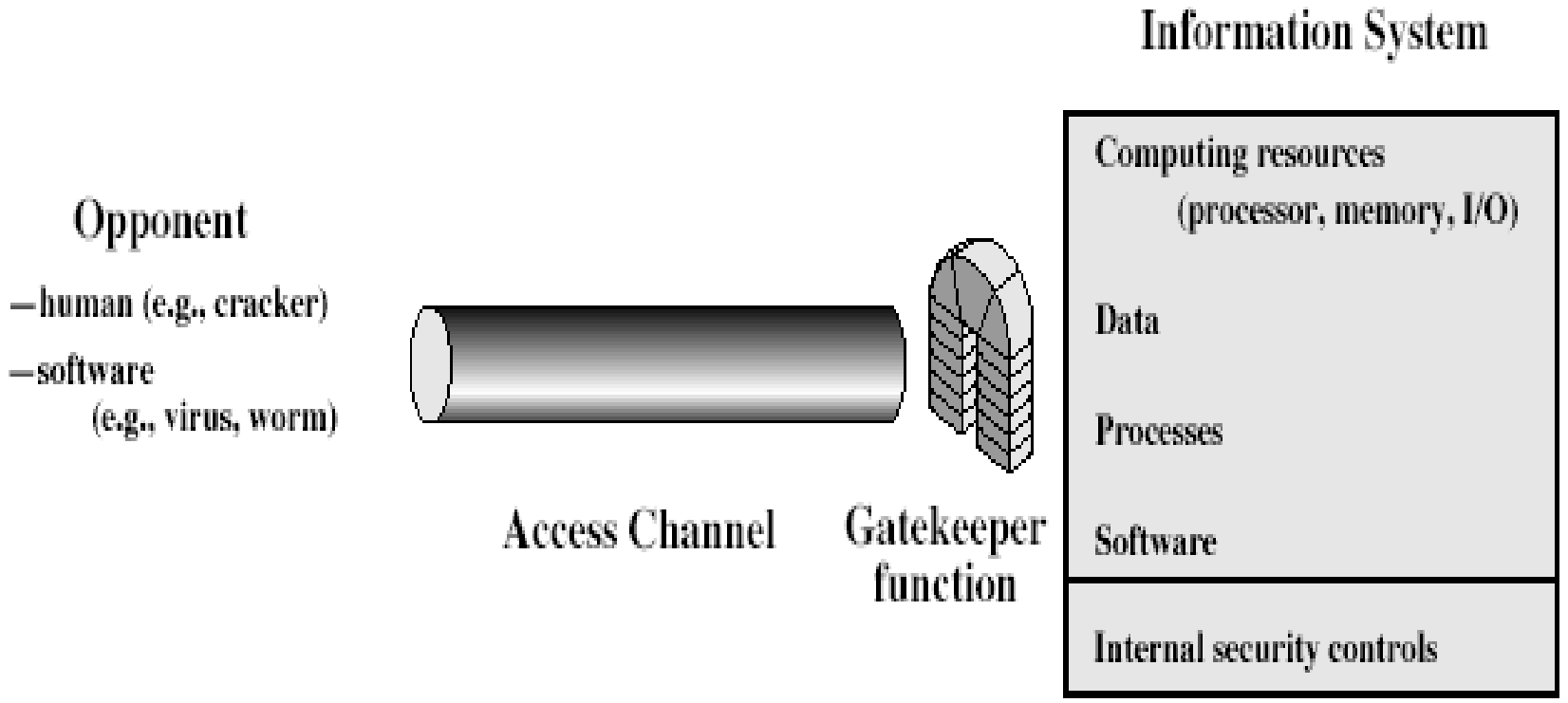


Model for Network Security

Using this model requires us to:

1. design a suitable algorithm for the security transformation (message de/encryption)
2. generate the secret information (keys) used by the algorithm
3. develop methods to distribute and share the secret information (keys)
4. specify a protocol enabling the principals to use the transformation and secret information for a security service (e.g. ssh)

Model for Network Access Security



- To prevent intruders enter into your system, you can introduce a gatekeeper program or firewall kind of thing before accessing your system.
- So opponents can not enter into your system to access data or software.
- There are two kinds of threads:
  - Information Access Threads
  - Service Threads

Internet standards and RFCs

- Internet security is responsible for the development and publication of standards for user over the internet.
- The Internet society is a professional membership organization that consists of number of boards and task forces involved in internet development and standardization.
- The internet society is the coordinating committee for internet design engineering an management.
- Three organizations in the Internet society are responsible for working standards development and publication.
  - Internet Architecture Board (IAB)
    - Defining overall Internet architecture
    - Providing guidance to IETF
  - Internet Engineering Task Force (IETF)
    - Actual development of protocols and standards
  - Internet Engineering Steering Group (IESG)
    - Technical management of IETF activities and Internet standards process
- RFCs (Request for Comments) are the working notes of the internet research and development community

Internet RFC Publication Standardization Process

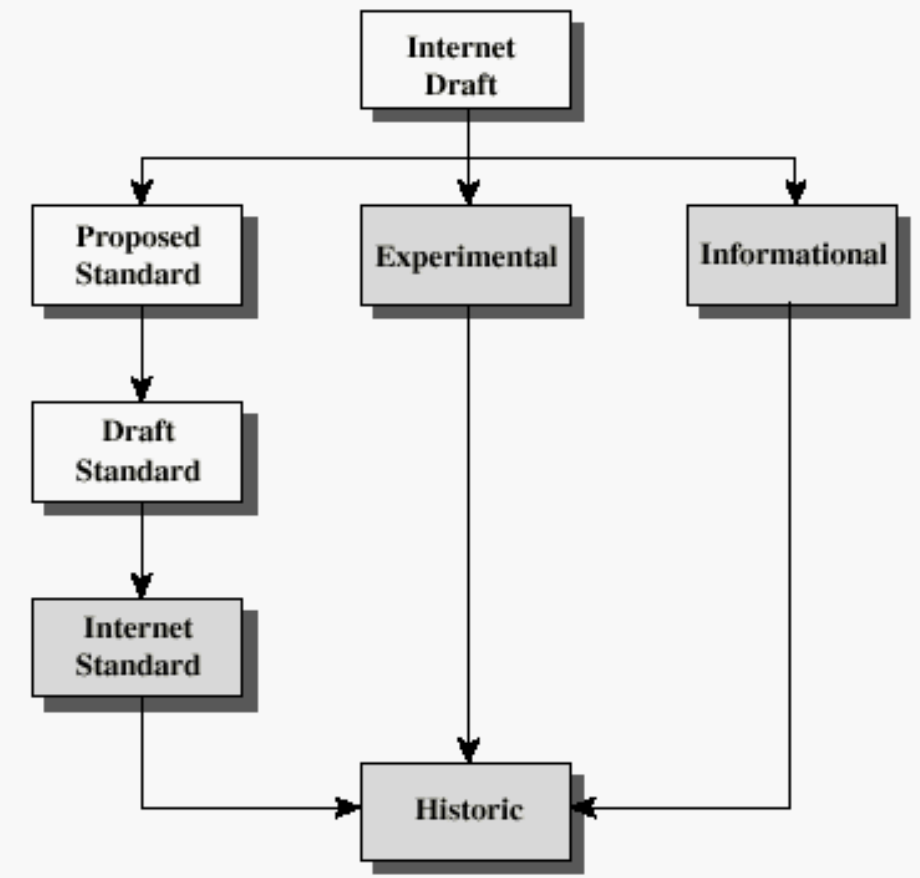


Figure 1.5 Internet RFC Publication Process

PART-III

- Symmetric Encryption principles
- Symmetric Block Encryption Algorithms
- Stream Ciphers and RC4
- Cipher Block modes of Operation
- Location of Encryption Devices
- Key Distribution

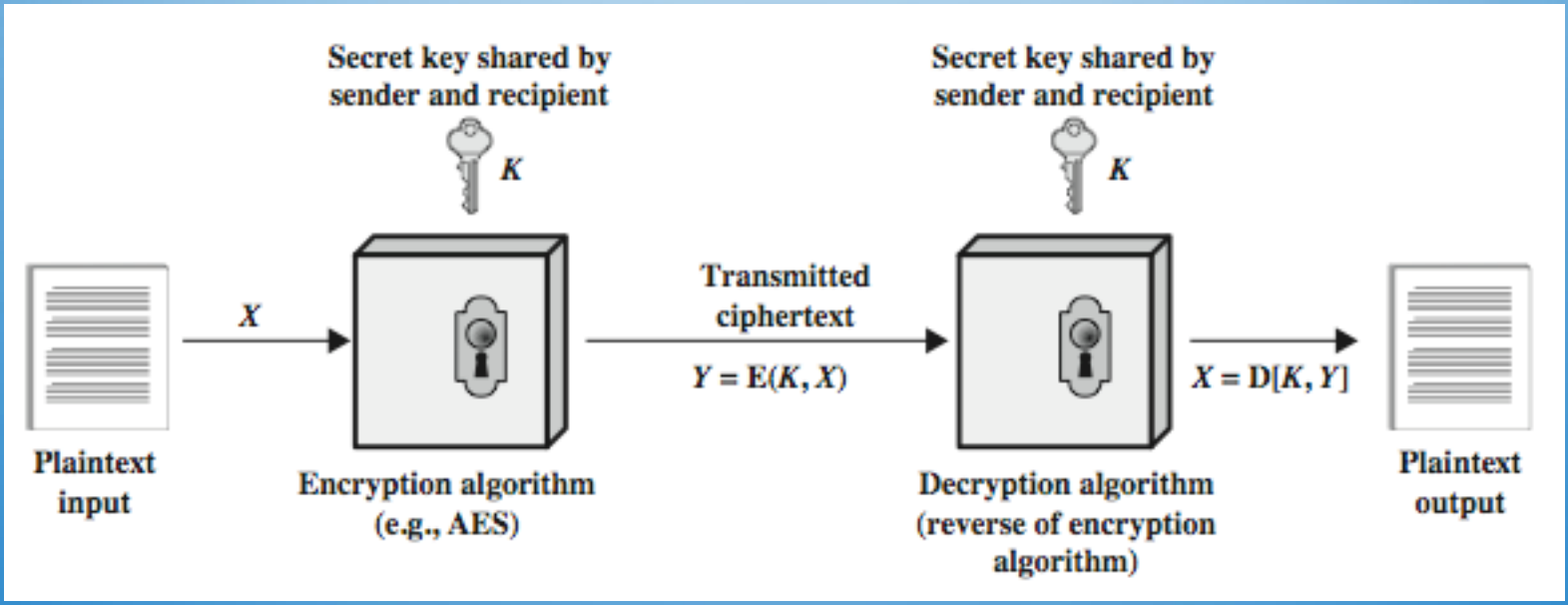
Symmetric Encryption principles

- Symmetric Encryption or conventional / private-key / single-key
- sender and recipient share a common key
- All classical encryption algorithms are private-key
- It was only type prior to invention of public-key in 1970's and by far most widely used
- **plaintext** - original message
- **ciphertext** - coded message
- **cipher** - algorithm for transforming plaintext to ciphertext
- **key** - info used in cipher known only to sender/receiver
- **encipher (encrypt)** - converting plaintext to ciphertext
- **decipher (decrypt)** - recovering plaintext from ciphertext

Cont…

- **cryptography** - study of encryption principles/methods
- **cryptology** - field of both cryptography and cryptanalysis
- **Cryptanalysis-** It is the Science of recovering plain text of the message without having access to the key

Symmetric Cipher Model



Cont…

- Two requirements for secure use of symmetric encryption:
  - a strong encryption algorithm
  - a secret key known only to sender / receiver
- Mathematically have:

  $Y = E(K, X)$

  $X = D(K, Y)$
- Assume encryption algorithm is known implies a secure channel to distribute key

Cryptography

- We can characterize cryptographic system by:
  - type of encryption operations used
    - substitution
    - transposition
    - product
  - number of keys used
    - single-key or private
    - two-key or public
  - way in which plaintext is processed
    - block
    - stream

Cont..

1. **The type of operations used for transforming plaintext to cipher text**.

All encryption algorithms are based on two general principles: substitution, in which each element in the plaintext (bit, letter, group of bits or letters) is mapped into another element, and transposition, in which elements in the plaintext are rearranged. The fundamental requirement is that no information be lost (that is, that all operations are reversible). Most systems, referred to as 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.

Cryptanalysis

- Objective to recover key not just message
- general approaches:
  - cryptanalytic attack
  - brute-force attack
- if either succeed all key use compromised

Cryptanalytic Attacks:

Cipher text only

only know algorithm & cipher text, is statistical, know or can identify plaintext

known plain text

know/suspect plaintext & cipher text

Chosen plaintext

select plaintext and obtain cipher text

Chosen cipher text

select cipher text and obtain plaintext

Chosen text select plaintext or cipher text to en/decrypt

Brute Force Search

- always possible to simply try every key
- most basic attack, proportional to key size
- assume either know / recognise plaintext

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

Cipher text only attack

- Given:

Ciphertext of many messages encrypted with same key

C1=Ek(P1), C2=Ek(P2), C3=Ek(P3)

- Task:

Find the plaintext of these message or even the key

Find P1, P2etc.. Or Key k

Known Plain text attack

- Given:

Cipher text and plain text of the corresponding message

P1,C1=Ek(P1),

P2,C2=Ek(P2),

P3,C3=Ek(P3)

- Task:

Find K

Or Algorithm to infer P_{i+1} from C_{i+1}

Chosen Plain Text Attack

- Given:

Plain Text and Ciphertext pairs

Can choose plaintext that gets encrypted

P1,C1=Ek(P1),

P2,C2=Ek(P2),

P3,C3=Ek(P3) Where P1,P2, P3 can be choosen

Task:

Find K

Or Algorithm to infer P_{i+1} from C_{i+1}

Chosen CipherText Attack

- Given:

Can choose different ciphertext to be decrypted

Access to decrypted Plaintext

C1, P1 =Dk(C1),

C2, P2 =Dk(C2),

C3, P3 =Dk(C3)

- Task:

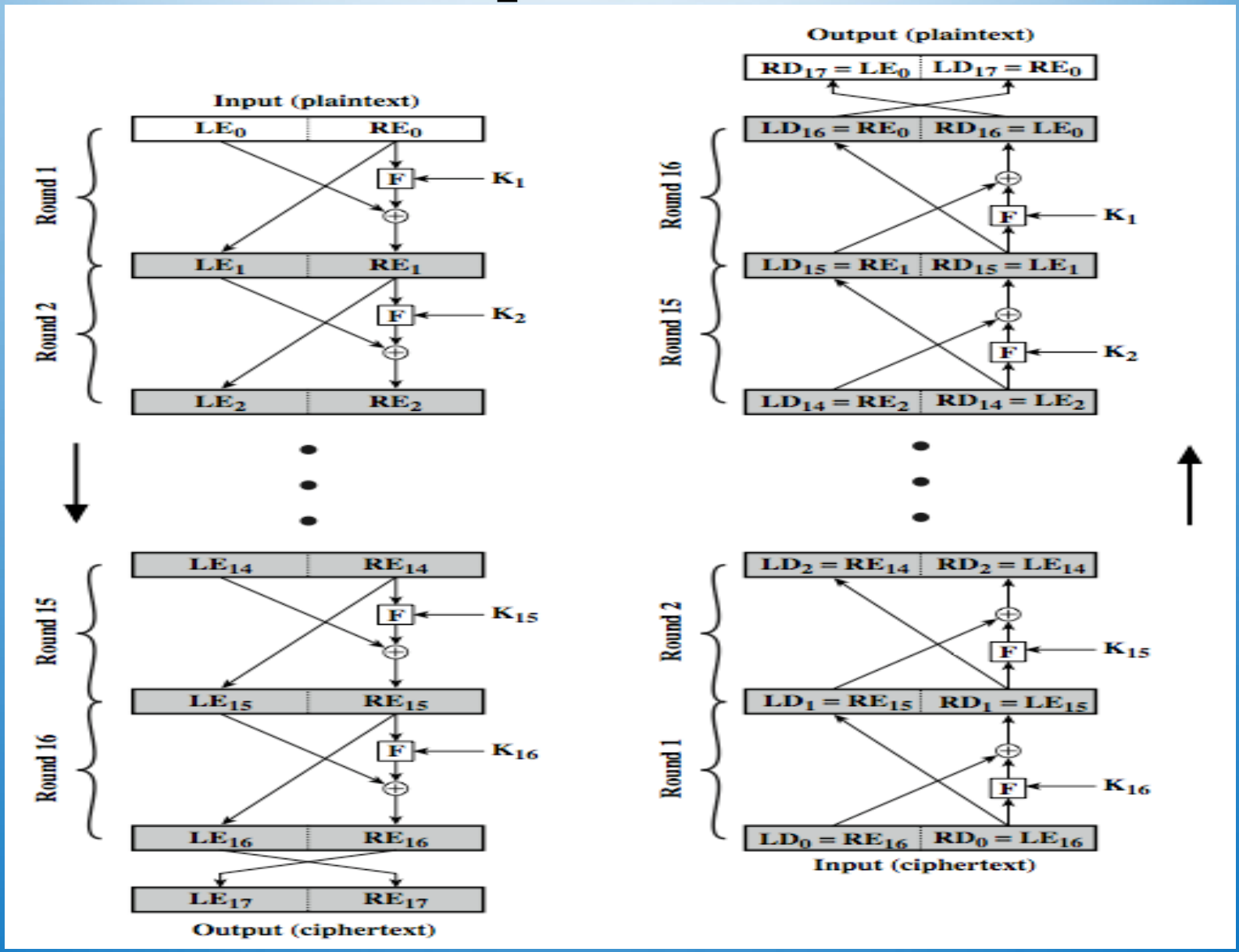
Find K

Note: Applicable in Public key Cryptography

Feistel Cipher Structure

- Horst Feistel devised the **Feistel cipher**
  - based on concept of invertible product cipher
- partitions input block into two halves
  - process through multiple rounds which
  - perform a substitution on left data half
  - based on round function of right half & subkey
  - then have permutation swapping halves
- implements Shannon's S-P net concept

Feistel Cipher Structure



Feistel Cipher Design Elements

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

The exact realization of a Feistel network depends on the choice of the following parameters and design features:

- Block size - increasing size improves security, but slows cipher
- key size - increasing size improves security, makes exhaustive key searching harder, but may slow cipher
- Number of rounds - increasing number improves security, but slows cipher
- Sub key generation algorithm - greater complexity can make analysis harder, but slows cipher
- Round function - greater complexity can make analysis harder, but slows cipher
- Fast software en/decryption - more recent concern for practical use
- Ease of analysis - for easier validation & testing of strength

DATA ENCRYPTION STANDARD (DES)

Outline

- History
- Encryption
- Key Generation
- Decryption
- Strength of DES
- Ultimate

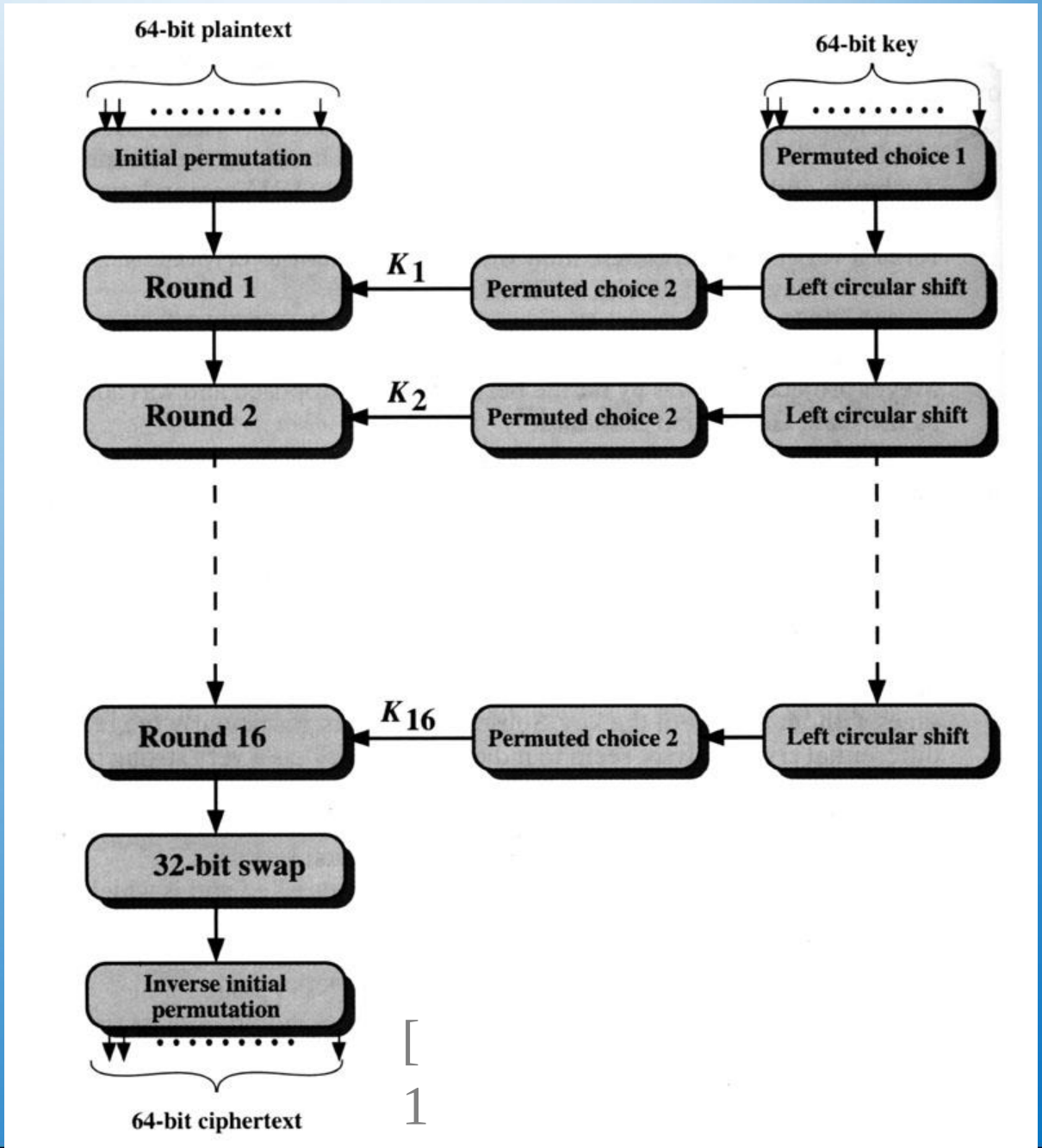
History

- In 1971, IBM developed an algorithm, named LUCIFER which operates on a block of 64 bits, using a 128-bit key

- Walter Tuchman, an IBM researcher, refined LUCIFER and reduced the key size to 56-bit, to fit on a chip.

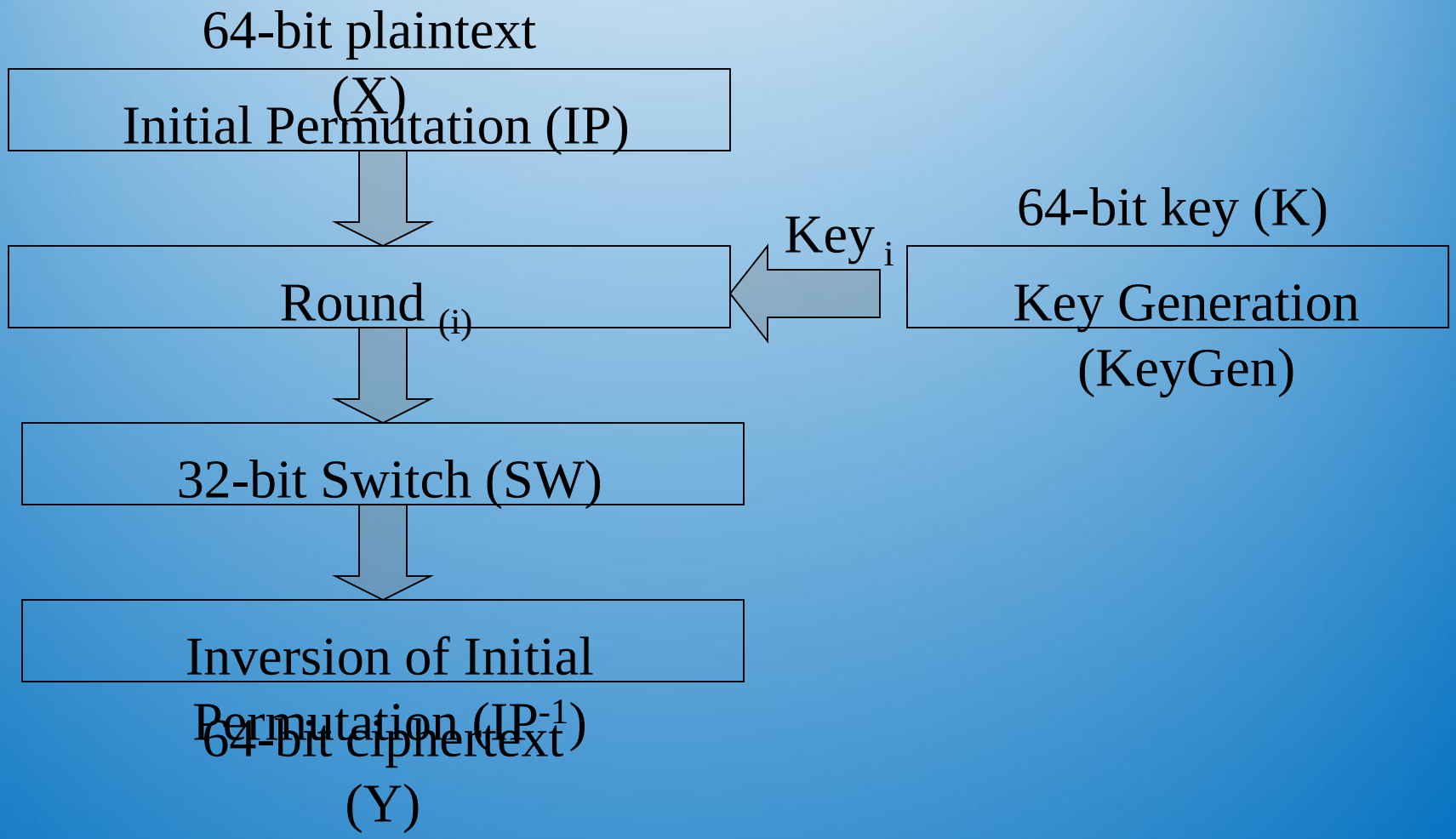
- In 1977, the results of Tuchman's project of IBM was adopted as the Data Encryption Standard by NSA (NIST).

Encryption



[
1

Encryption (cont.)



Encryption (cont.)

- Plaintext: X
- Initial Permutation: IP( )
- $Round_i$: $1 \leq i \leq 16$
- 32-bit switch: SW( )
- Inverse IP: IP^{-1}( )
- Ciphertext: Y
- $Y = IP^{-1}(SW(Round_i(IP(X), Key_i)))$

Encryption (IP, IP^{-1})

- IP

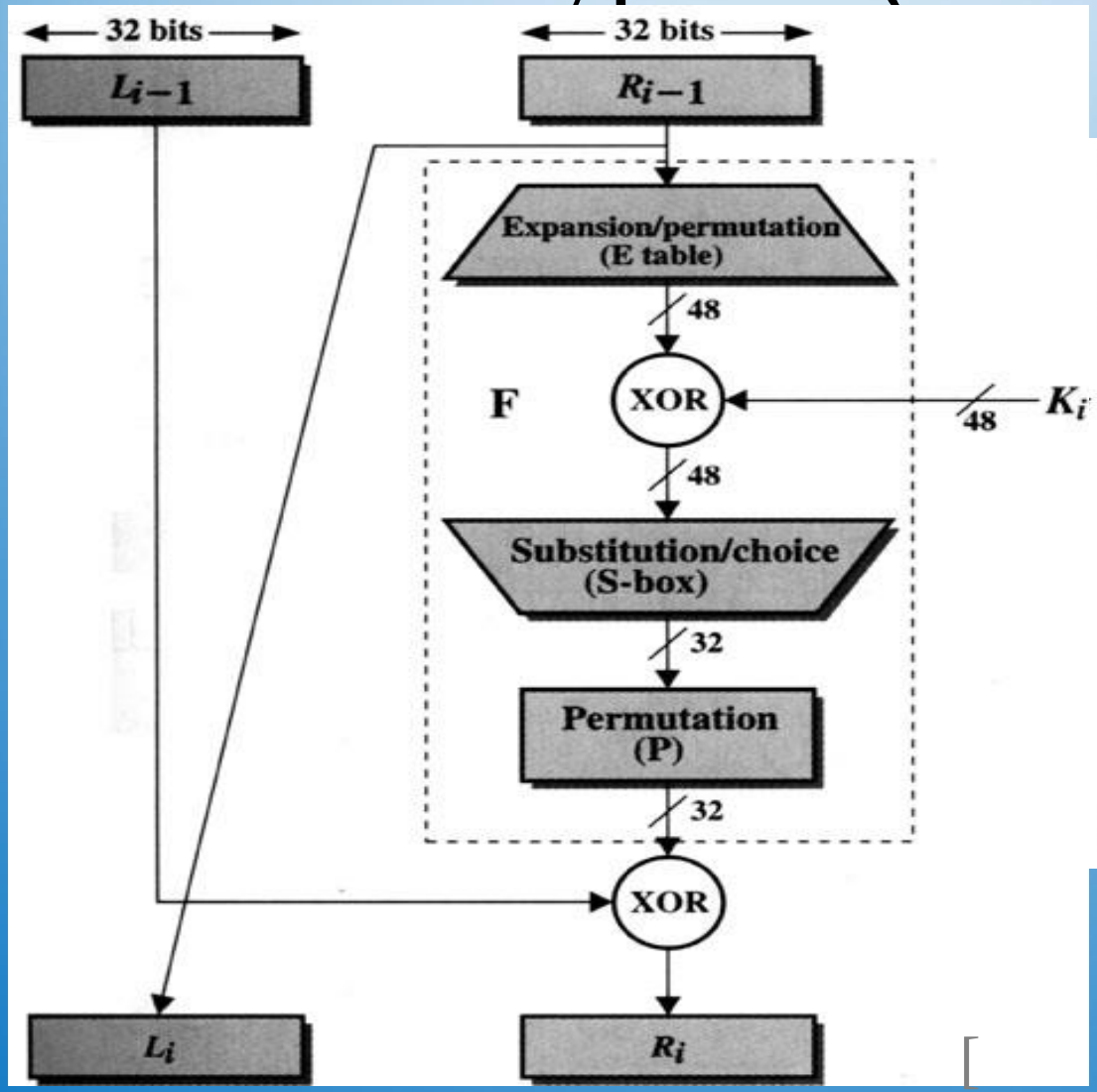
Bit	0	1	2	3	4	5	6	7
1	58	50	42	34	26	18	10	2
9	60	52	44	36	28	20	12	4
17	62	54	46	38	30	22	14	6
25	64	56	48	40	32	24	16	8
33	57	49	41	33	25	17	9	1
41	59	51	43	35	27	19	11	3
49	61	53	45	37	29	21	13	5
57	63	55	47	39	31	23	15	7

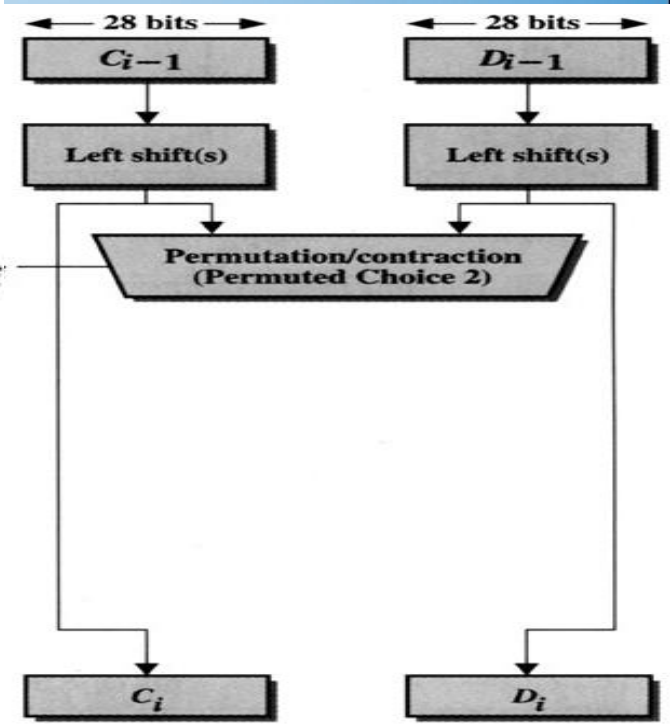
- IP^{-1}

Bit	0	1	2	3	4	5	6	7
1	40	8	48	16	56	24	64	32
9	39	7	47	15	55	23	63	31
17	38	6	46	14	54	22	62	30
25	37	5	45	13	53	21	61	29
33	36	4	44	12	52	20	60	28
41	35	3	43	11	51	19	59	27
49	34	2	42	10	50	18	58	26
57	33	1	41	9	49	17	57	25

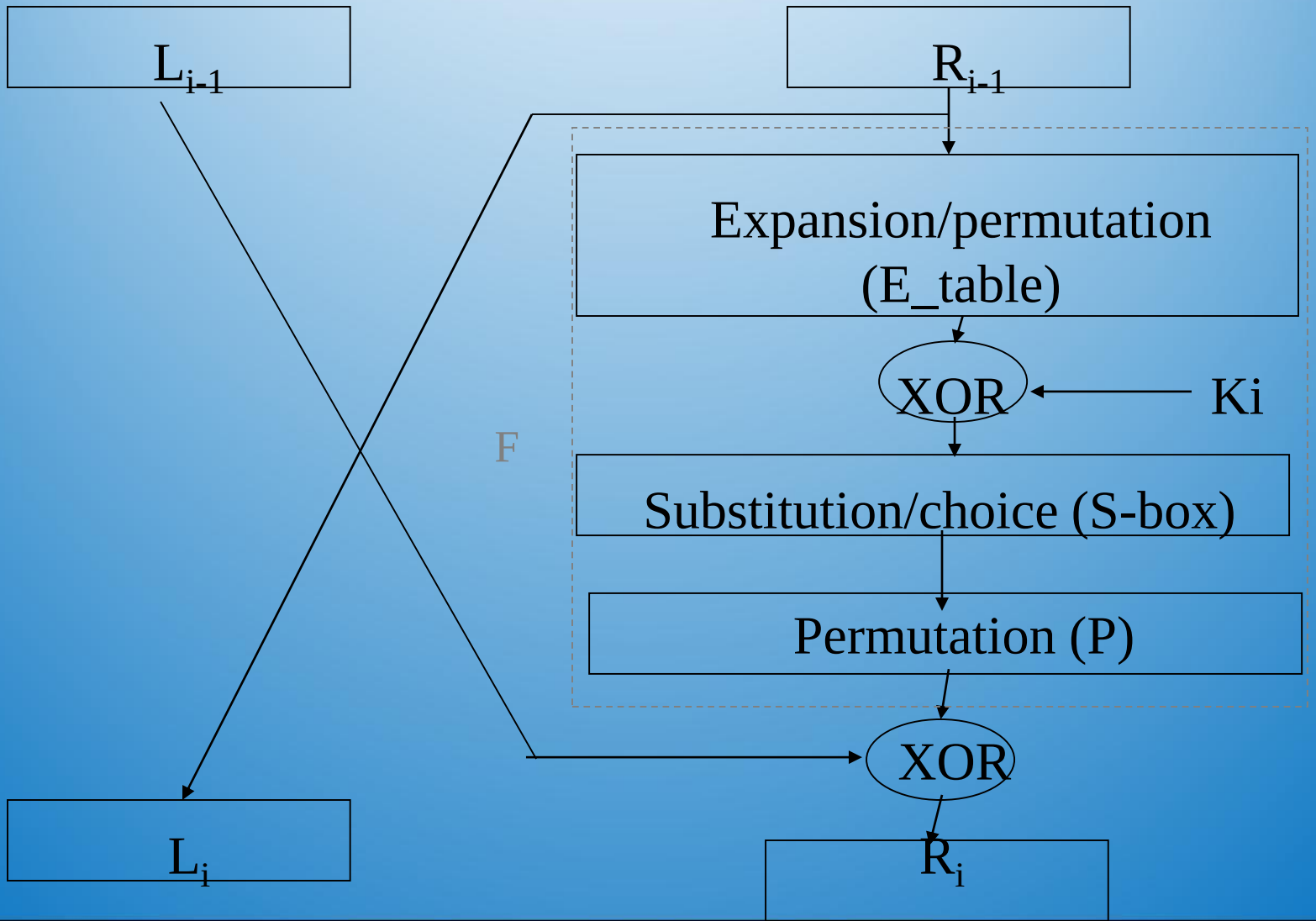
- Note: $IP(IP^{-1}) = IP^{-1}(IP) = I$

Encryption (Round) (Key Generation)



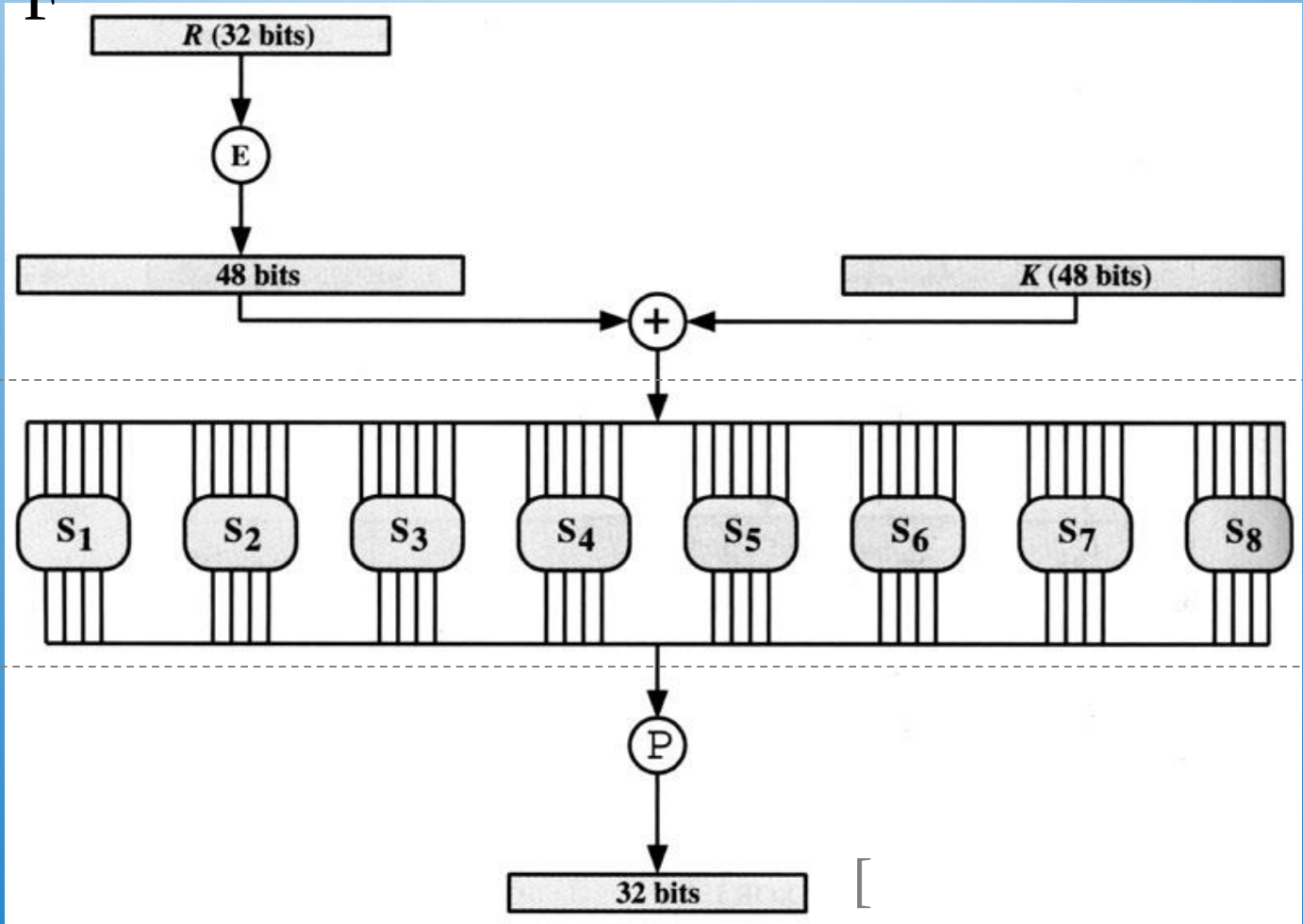


Encryption (Round) (cont.)



Encryption (Round) (cont.)

F



[

1

Encryption (Round) (cont.)

- Separate plaintext as L_0R_0
  - L_0: left half 32 bits of plaintext
  - R_0: right half 32 bits of plaintext
- Expansion/permutation: E( )
- Substitution/choice: S-box( )
- Permutation: P( )

} F

- $R_i = L_{i-1} \sim P(S\_box(E(R_{i-1}) \sim Key_i))$
- $L_i = R_{i-1}$

Encryption (Round) (cont.)

- E

32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	45	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

Expansion

- 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
9	13	30	6	22	11	4	25

Expansion

Encryption (Round) (cont.)

- S-box

S_1

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

S_2

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

S_3

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

S_4

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

S_5

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

S_6

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

S_7

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

S_8

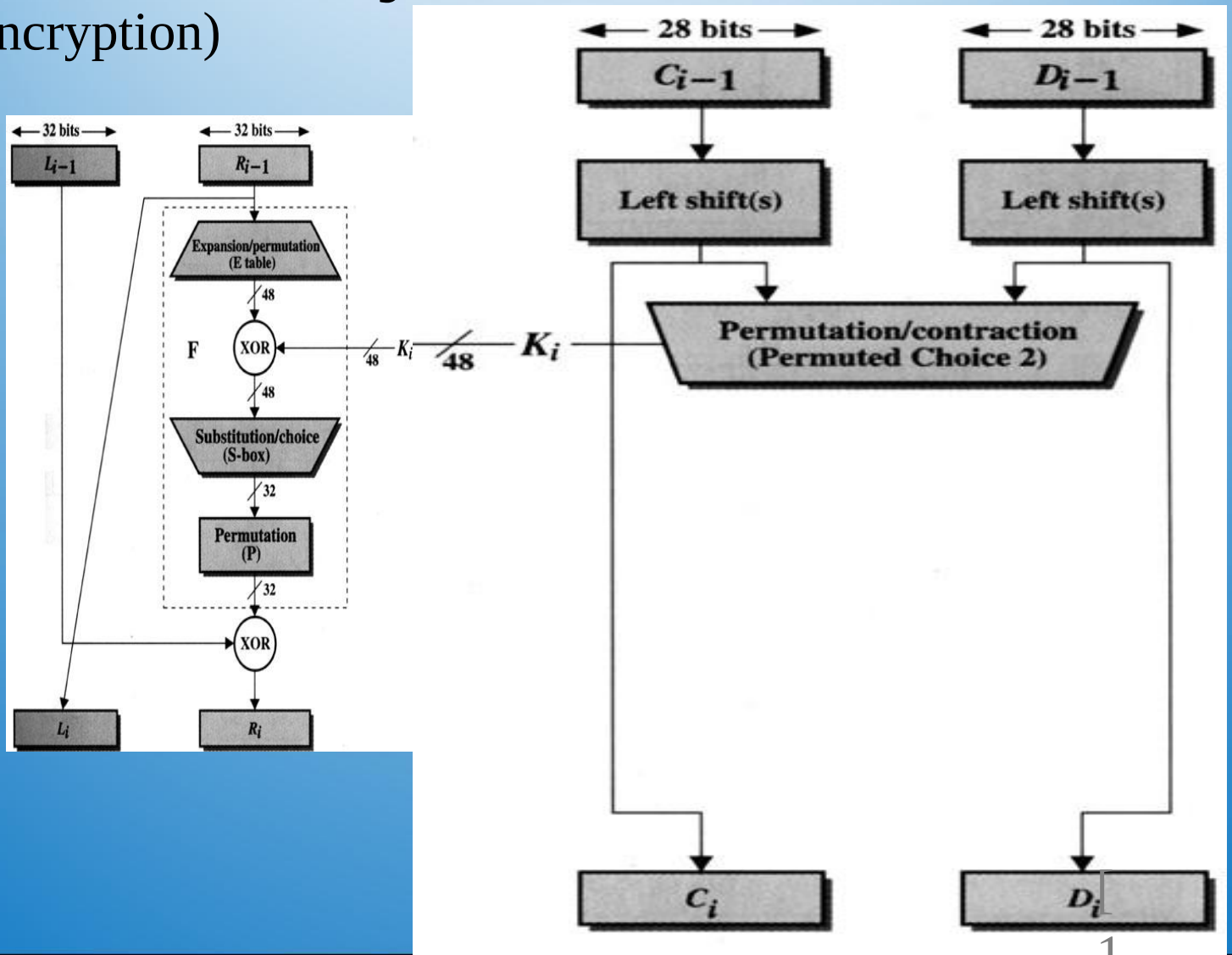
13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

- This table lists the eight S-boxes used in DES.
- Each S-box replaces a 6-bit input with a 4-bit output.
- Given a 6-bit input, the 4-bit output is found by selecting the row using the outer two bits, and the column using the inner four bits.

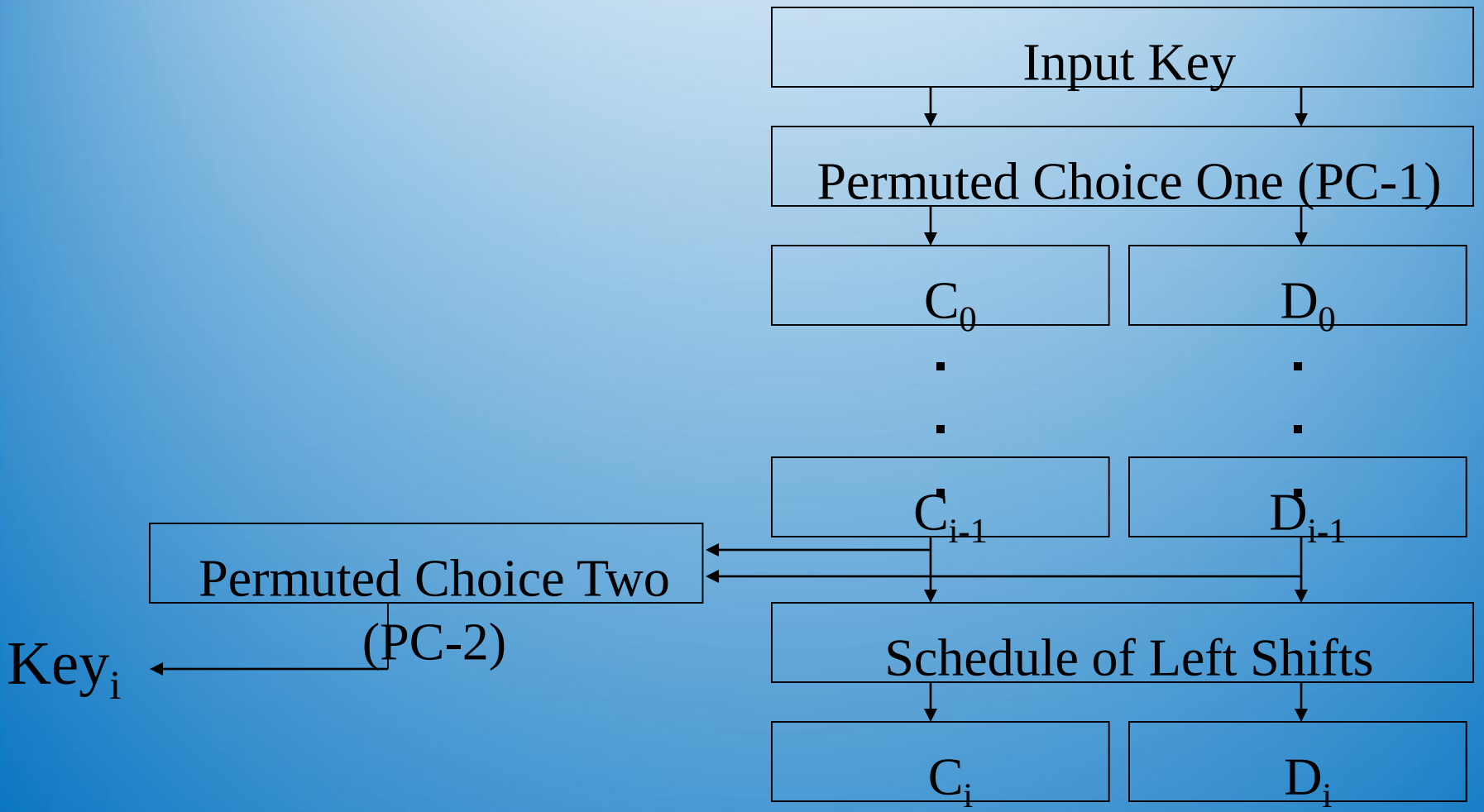
For example, an input "**0**11011**1**" has outer bits "**01**" and inner bits "1101"; noting that the first row is "00" and the first column is "0000", the corresponding output for S-box S_5 would be "1001" (=9), the value in the second row, 14th column.

Key Generation

(Encryption)



Key Generation (cont.)



Key Generation (cont.)

- Original Key: Key_0
- Permuted Choice One: PC_1( )
- Permuted Choice Two: PC_2( )
- Schedule of Left Shift: SLS( )
- $(C_0, D_0) = PC\_1(Key_0)$
- $(C_i, D_i) = SLS(C_{i-1}, D_{i-1})$
- $Key_i = PC\_2(SLS(C_{i-1}, D_{i-1}))$

Permuted choice 1 (PC-1)

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

Permuted choice 2 (PC-2)

PC-2

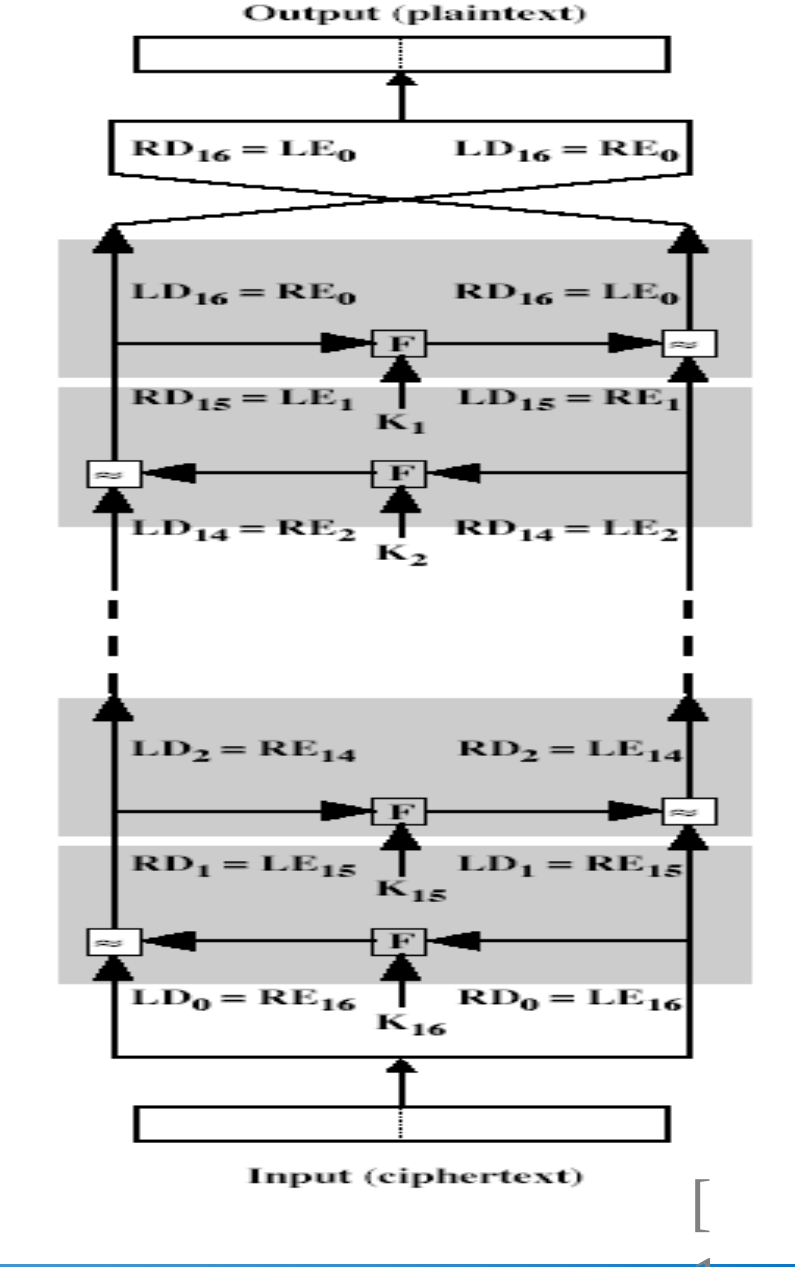
14	17	11	24	1	5
3	28	15	6	21	10
23	19	12	4	26	8
16	7	27	20	13	2
41	52	31	37	47	55
30	40	51	45	33	48
44	49	39	56	34	53
46	42	50	36	29	32

Rotations in the key-schedule

Round number	Number of left rotations
1	1
2	1
3	2
4	2
5	2
6	2
7	2
8	2
9	1
10	2
11	2
12	2
13	2
14	2
15	2
16	1

Decrypt

- The same algorithm as encryption.
- Reversed the order of key (Key_{16}, Key_{15}, … Key_1).
- For example:
  - IP undoes IP^{-1} step of encryption.
  - 1st round with SK16 undoes 16th encrypt round.



Strength of DES

- Criticism
  - Reduction in key size of 72 bits
    - Too short to withstand with brute-force attack
  - S-boxes were classified.
    - Weak points enable NSA to decipher without key.
- 56-bit keys have $2^{56} = 7.2 \times 10^{16}$ values
  - Brute force search looks hard.
  - A machine performing one DES encryption per microsecond would take more than a thousand year to break the cipher.

Strength of DES (cont.)

- Avalanche effect in DES
  - If a small change in either the plaintext or the key, the ciphertext should change markedly.
- DES exhibits a strong avalanche effect.

(a) Change in Plaintext		(b) Change in Key	
Round	Number of bits that differ	Round	Number of bits that differ
0	1	0	0
1	6	1	2
2	21	2	14
3	35	3	28
4	39	4	32
5	34	5	30
6	32	6	32
7	31	7	35
8	29	8	34
9	42	9	40
10	44	10	38
11	32	11	31
12	30	12	33
13	30	13	28
14	26	14	26
15	29	15	34
16	34	16	35

Ultimate

- DES was proved insecure
  - In 1997 on Internet in a few months
  - in 1998 on dedicated h/w (EFF) in a few days
  - In 1999 above combined in 22hrs!

Multiple Encryption & DES

- clear a replacement for DES was needed
  - theoretical attacks that can break it
  - demonstrated exhaustive key search attacks
- AES is a new cipher alternative
- prior to this alternative was to use multiple encryption with DES implementations
- Triple-DES is the chosen form

Double-DES?

- could use 2 DES encrypts on each block
  - $C = E_{K2}(E_{K1}(P))$
- issue of reduction to single stage
- and have "meet-in-the-middle" attack
  - works whenever use a cipher twice
  - since $X = E_{K1}(P) = D_{K2}(C)$
  - attack by encrypting P with all keys and store
  - then decrypt C with keys and match X value
  - can show takes $O(2^{56})$ steps

Triple-DES with Two-Keys

- hence must use 3 encryptions
  - would seem to need 3 distinct keys
- but can use 2 keys with E-D-E sequence
  - $C = E_{K1}(D_{K2}(E_{K1}(P)))$
  - nb encrypt & decrypt equivalent in security
  - if K1=K2 then can work with single DES
- standardized in ANSI X9.17 & ISO8732
- no current known practical attacks
  - several proposed impractical attacks might become basis of future attacks

Triple-DES with Three-Keys

- although are no practical attacks on two-key Triple-DES have some indications
- can use Triple-DES with Three-Keys to avoid even these
  - $C = E_{K3}(D_{K2}(E_{K1}(P)))$
- has been adopted by some Internet applications, eg PGP, S/MIME

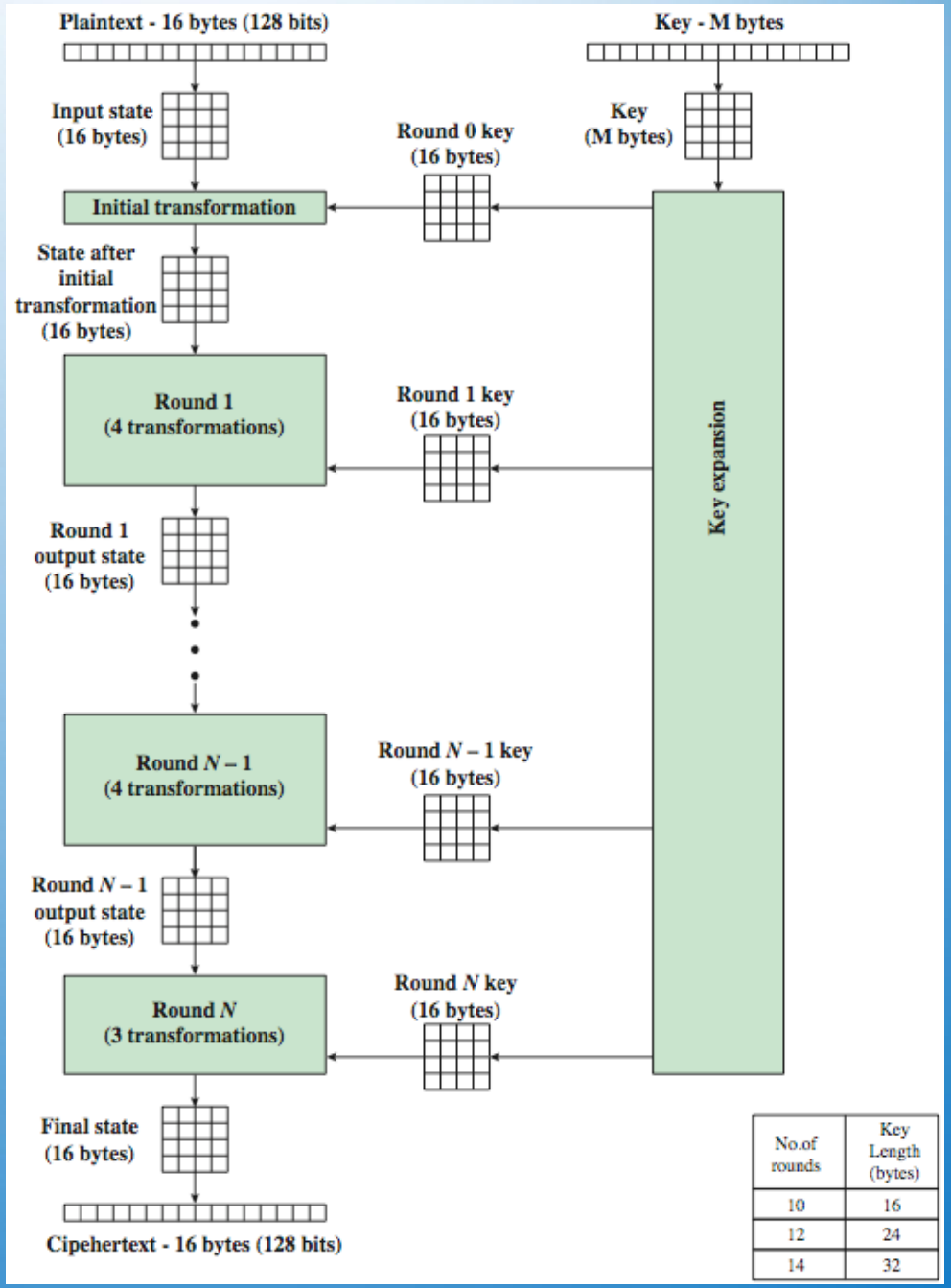
AES Origins

- clear a replacement for DES was needed
  - have theoretical attacks that can break it
  - have demonstrated exhaustive key search attacks
- can use Triple-DES – but slow, has small blocks
- US NIST issued call for ciphers in 1997
- 15 candidates accepted in Jun 98
- 5 were shortlisted in Aug-99
- Rijndael was selected as the AES in Oct-2000
- issued as FIPS PUB 197 standard in Nov-2001

The AES Cipher - Rijndael

- designed by Rijmen-Daemen in Belgium
- has 128/192/256 bit keys, 128 bit data
- an **iterative** rather than **Feistel** cipher
  - processes data as block of 4 columns of 4 bytes
  - operates on entire data block in every round
- designed to have:
  - resistance against known attacks
  - speed and code compactness on many CPUs
  - design simplicity

AES Encryption Process

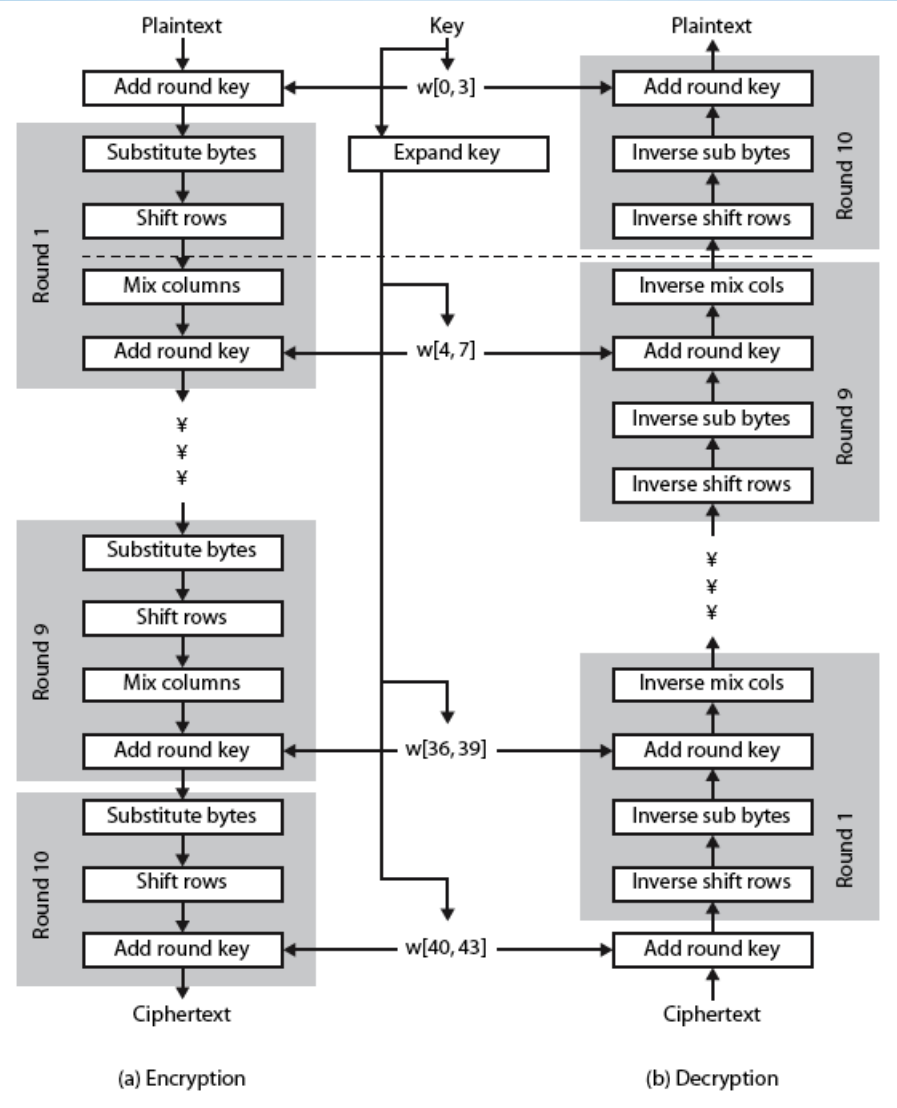


No.of rounds	Key Length (bytes)
10	16
12	24
14	32

AES Structure

- data block of 4 columns of 4 bytes is state
- key is expanded to array of words
- has 9/11/13 rounds in which state undergoes:
  - byte substitution (1 S-box used on every byte)
  - shift rows (permute bytes between groups/columns)
  - mix columns (subs using matrix multiply of groups)
  - add round key (XOR state with key material)
  - view as alternating XOR key & scramble data bytes
- initial XOR key material & incomplete last round
- with fast XOR & table lookup implementation

AES Structure



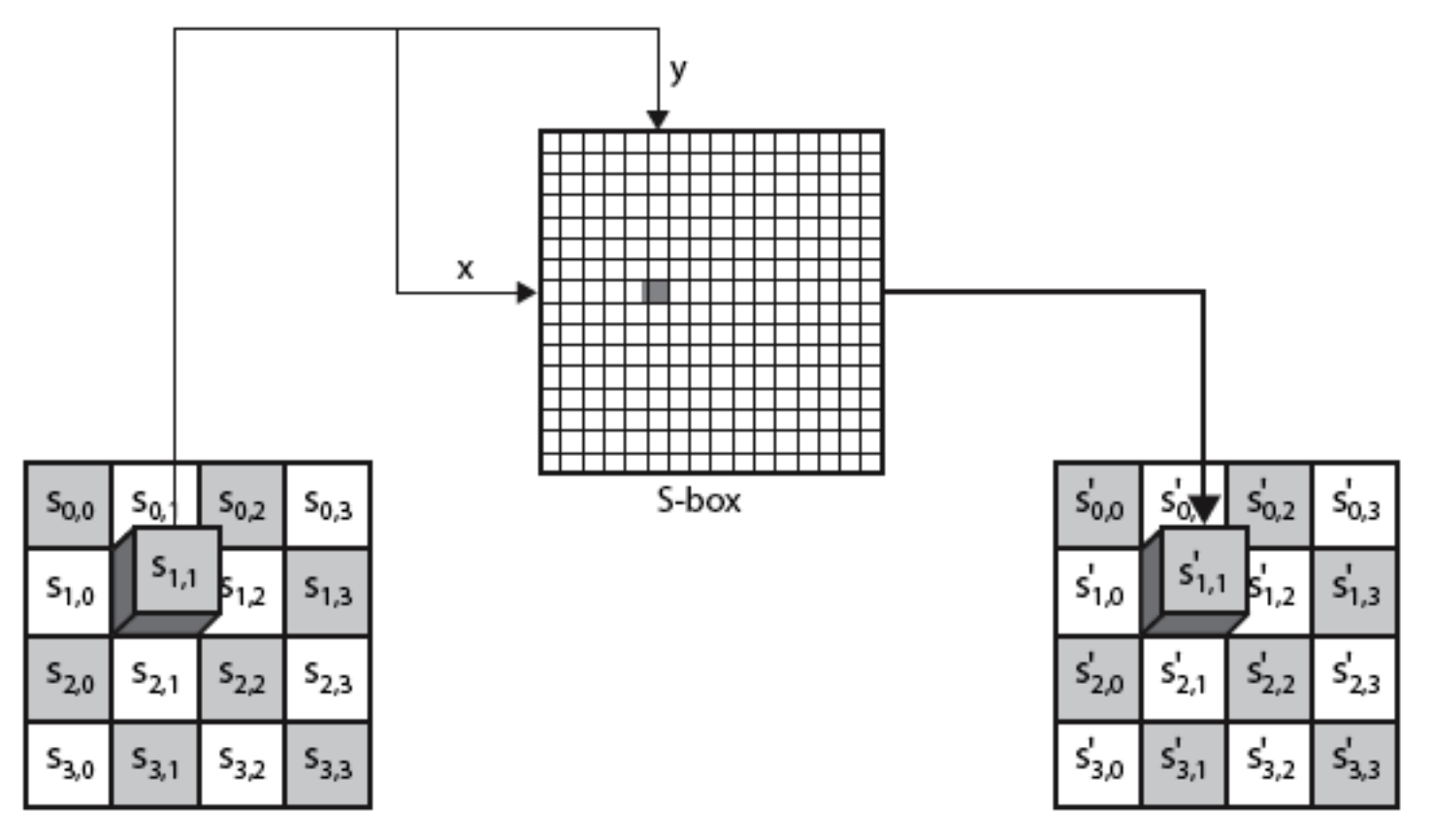
Some Comments on AES

1. an **iterative** rather than **Feistel** cipher
2. key expanded into array of 32-bit words
    1. four words form round key in each round
3. 4 different stages are used as shown
4. has a simple structure
5. only AddRoundKey uses key
6. AddRoundKey a form of Vernam cipher
7. each stage is easily reversible
8. decryption uses keys in reverse order
9. decryption does recover plaintext
10. final round has only 3 stages

Substitute Bytes

- a simple substitution of each byte
- uses one table of 16x16 bytes containing a permutation of all 256 8-bit values
- each byte of state is replaced by byte indexed by row (left 4-bits) & column (right 4-bits)
  - eg. byte {95} is replaced by byte in row 9 column 5
  - which has value {2A}
- S-box constructed using defined transformation of values in GF(2^8)
- designed to be resistant to all known attacks

Substitute Bytes



Substitute Bytes Example

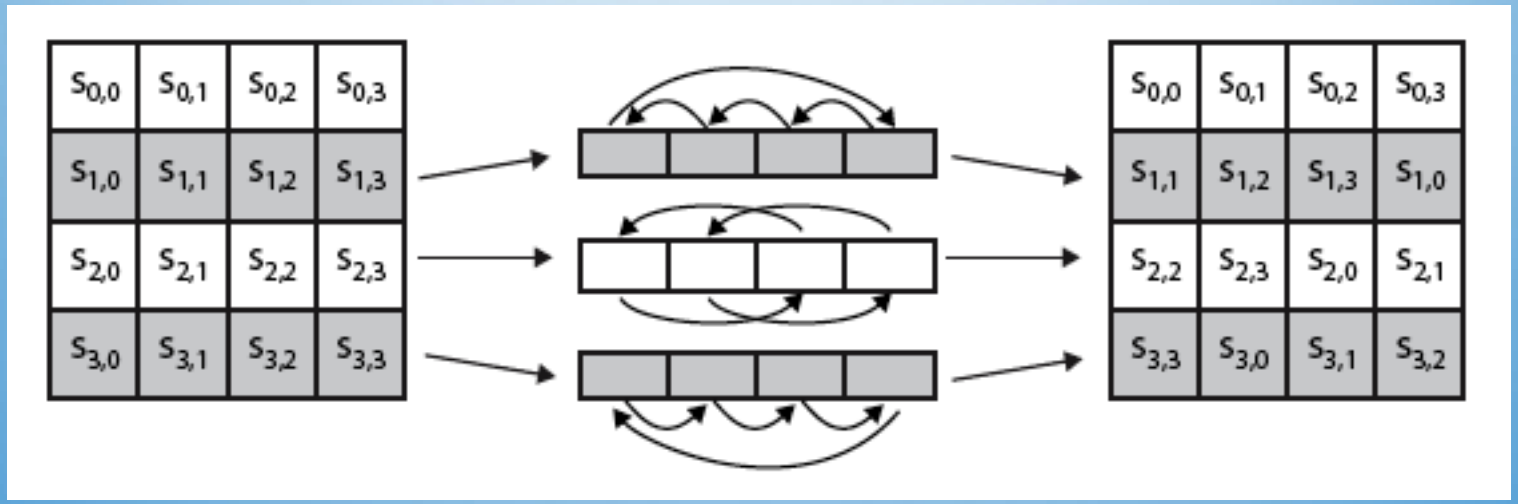
EA	04	65	85
83	45	5D	96
5C	33	98	B0
F0	2D	AD	C5

87	F2	4D	97
EC	6E	4C	90
4A	C3	46	E7
8C	D8	95	A6

Shift Rows

- a circular byte shift in each each
  - 1st row is unchanged
  - 2nd row does 1 byte circular shift to left
  - 3rd row does 2 byte circular shift to left
  - 4th row does 3 byte circular shift to left
- decrypt inverts using shifts to right
- since state is processed by columns, this step permutes bytes between the columns

Shift Rows



87	F2	4D	97
EC	6E	4C	90
4A	C3	46	E7
8C	D8	95	A6

→

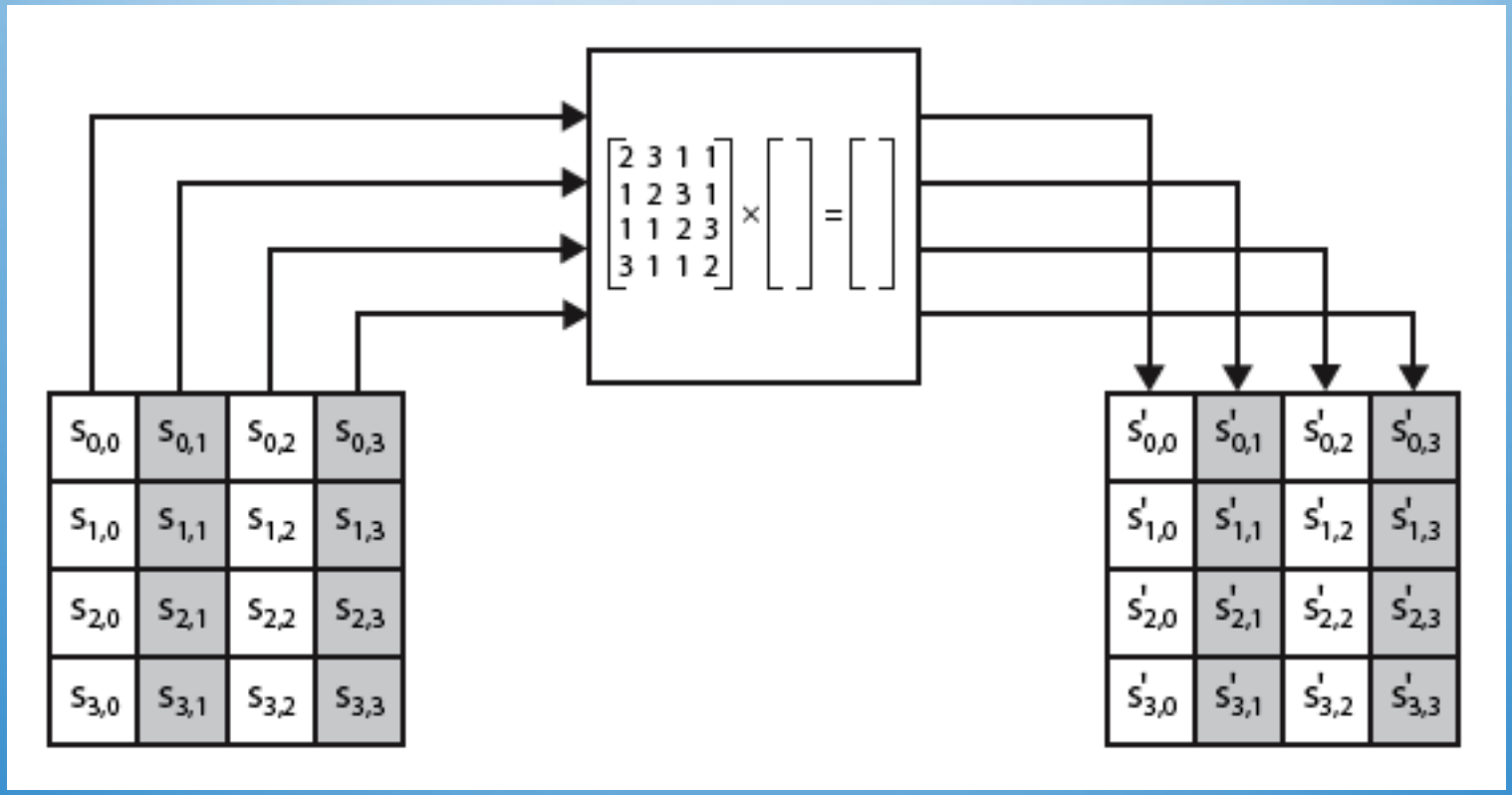
87	F2	4D	97
6E	4C	90	EC
46	E7	4A	C3
A6	8C	D8	95

Mix Columns

- each column is processed separately
- each byte is replaced by a value dependent on all 4 bytes in the column
- effectively a matrix multiplication in GF(2^8) using prime poly $m(x) = x^8+x^4+x^3+x+1$

$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{bmatrix} = \begin{bmatrix} s'_{0,0} & s'_{0,1} & s'_{0,2} & s'_{0,3} \\ s'_{1,0} & s'_{1,1} & s'_{1,2} & s'_{1,3} \\ s'_{2,0} & s'_{2,1} & s'_{2,2} & s'_{2,3} \\ s'_{3,0} & s'_{3,1} & s'_{3,2} & s'_{3,3} \end{bmatrix}$$

Mix Columns


2 3 1 1
1 2 3 1
1 1 2 3
3 1 1 2
×
=
$s_{0,0}$ $s_{0,1}$ $s_{0,2}$ $s_{0,3}$
$s_{1,0}$ $s_{1,1}$ $s_{1,2}$ $s_{1,3}$
$s_{2,0}$ $s_{2,1}$ $s_{2,2}$ $s_{2,3}$
$s_{3,0}$ $s_{3,1}$ $s_{3,2}$ $s_{3,3}$
$s'_{0,0}$ $s'_{0,1}$ $s'_{0,2}$ $s'_{0,3}$
$s'_{1,0}$ $s'_{1,1}$ $s'_{1,2}$ $s'_{1,3}$
$s'_{2,0}$ $s'_{2,1}$ $s'_{2,2}$ $s'_{2,3}$
$s'_{3,0}$ $s'_{3,1}$ $s'_{3,2}$ $s'_{3,3}$

Mix Columns Example

87	F2	4D	97
6E	4C	90	EC
46	E7	4A	C3
A6	8C	D8	95

→

47	40	A3	4C
37	D4	70	9F
94	E4	3A	42
ED	A5	A6	BC

$$(\{02\} \bullet \{87\}) \oplus (\{03\} \bullet \{6E\}) \oplus \{46\} \oplus \{A6\} = \{47\}$$

$$\{87\} \oplus (\{02\} \bullet \{6E\}) \oplus (\{03\} \bullet \{46\}) \oplus \{A6\} = \{37\}$$

$$\{87\} \oplus \{6E\} \oplus (\{02\} \bullet \{46\}) \oplus (\{03\} \bullet \{A6\}) = \{94\}$$

$$(\{03\} \bullet \{87\}) \oplus \{6E\} \oplus \{46\} \oplus (\{02\} \bullet \{A6\}) = \{ED\}$$

AES Arithmetic

- uses arithmetic in the finite field GF(2^8)
- with irreducible polynomial

$m(x) = x^8 + x^4 + x^3 + x + 1$

which is (100011011) or {11b}

- e.g.

{02} • {87} mod {11b} = (1 0000 1110) mod {11b}

= (1 0000 1110) xor (1 0001 1011) = (0001 0101)

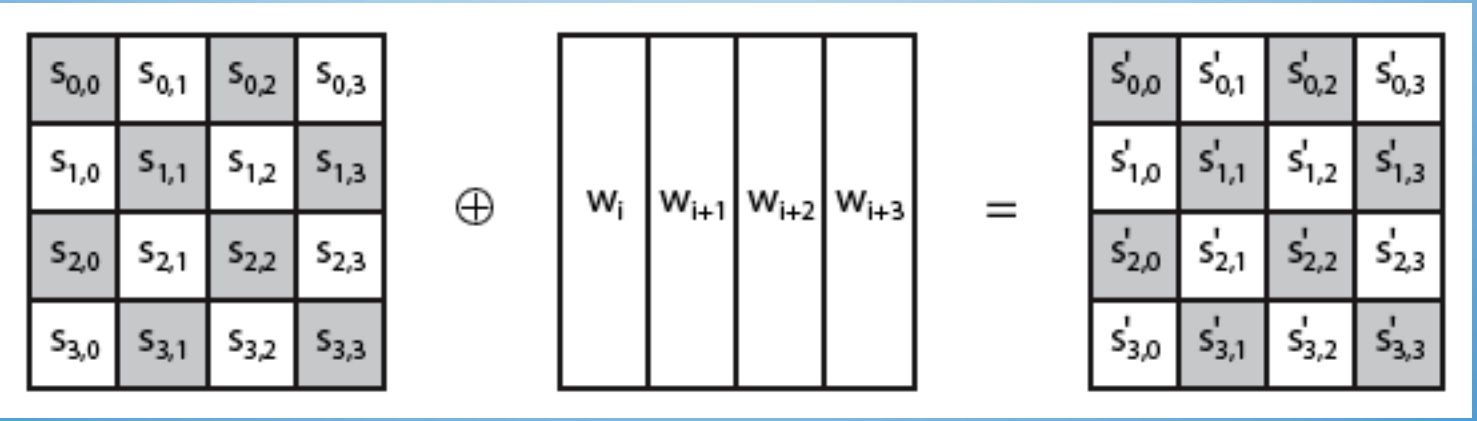
Mix Columns

- can express each col as 4 equations
  - to derive each new byte in col
- decryption requires use of inverse matrix
  - with larger coefficients, hence a little harder
- have an alternate characterisation
  - each column a 4-term polynomial
  - with coefficients in GF(2^8)
  - and polynomials multiplied modulo (x^4+1)
- coefficients based on linear code with maximal distance between codewords

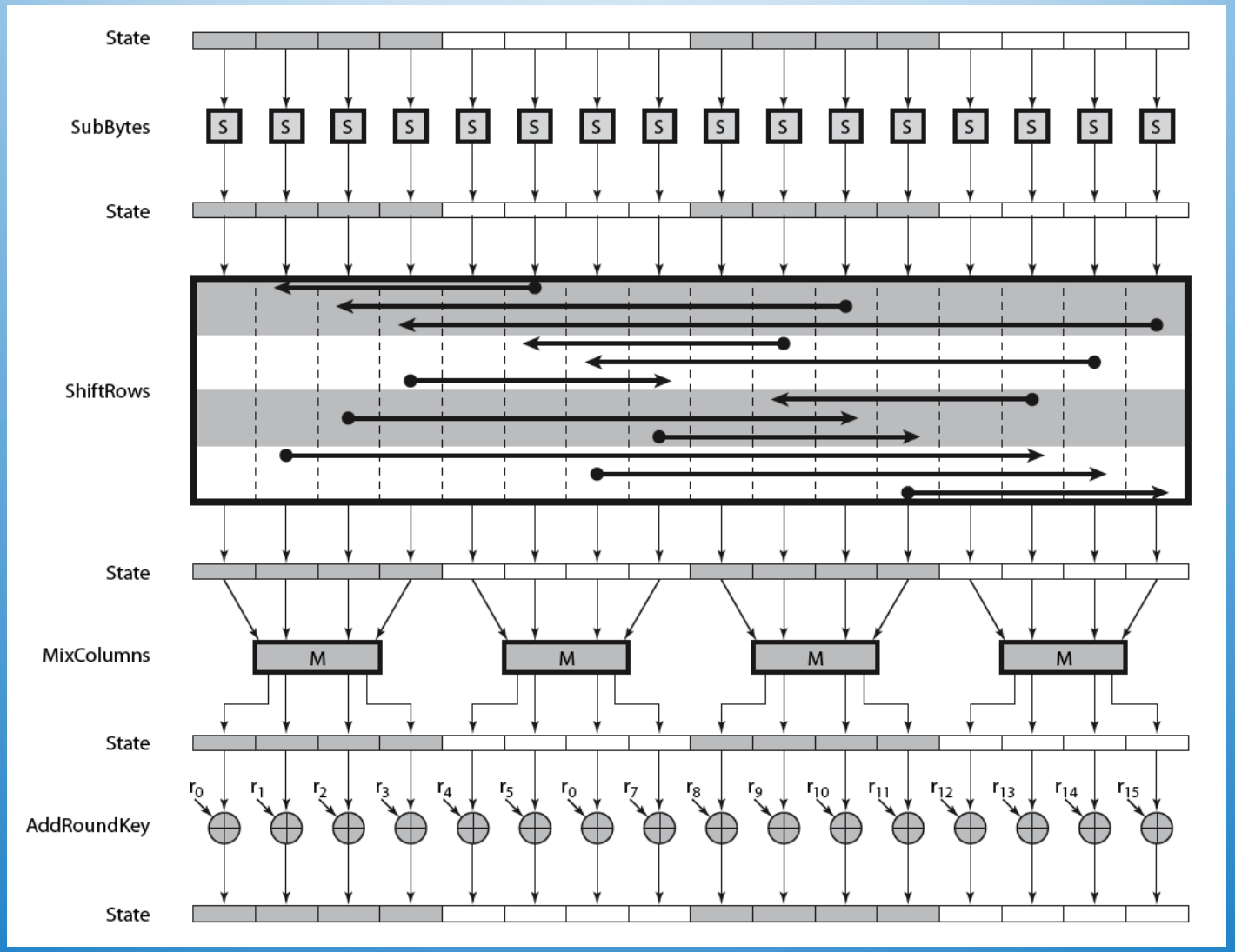
Add Round Key

- XOR state with 128-bits of the round key
- again processed by column (though effectively a series of byte operations)
- inverse for decryption identical
  - since XOR own inverse, with reversed keys
- designed to be as simple as possible
  - a form of Vernam cipher on expanded key
  - requires other stages for complexity / security

Add Round Key



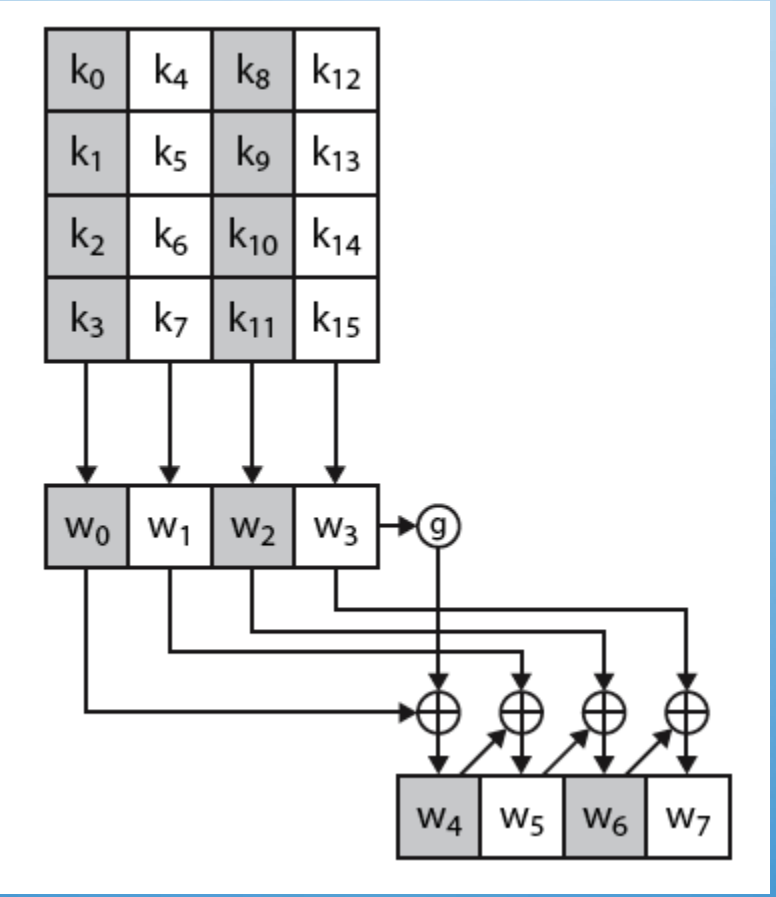
AES Round


State
SubBytes
S
State
ShiftRows
State
MixColumns
M
State
AddRoundKey
r0 r1 r2 r3 r4 r5 r0 r7 r8 r9 r10 r11 r12 r13 r14 r15
State

AES Key Expansion

- takes 128-bit (16-byte) key and expands into array of 44/52/60 32-bit words
- start by copying key into first 4 words
- then loop creating words that depend on values in previous & 4 places back
  - in 3 of 4 cases just XOR these together
  - 1st word in 4 has rotate + S-box + XOR round constant on previous, before XOR 4th back

AES Key Expansion



Key Expansion Rationale

- designed to resist known attacks
- design criteria included
  - knowing part key insufficient to find many more
  - invertible transformation
  - fast on wide range of CPU's
  - use round constants to break symmetry
  - diffuse key bits into round keys
  - enough non-linearity to hinder analysis
  - simplicity of description

One-Time Pad

- Developed by Gilbert Vernam in 1918, another name: ***Vernam Cipher***
- The key
  - a truly random sequence of 0's and 1's
  - the same length as the message
  - use one time only
- The encryption
  - adding the key to the message modulo 2, bit by bit.

Encryption $c_i = m_i \oplus k_i \qquad i = 1,2,3,\ldots$

Decryption $m_i = c_i \oplus k_i \qquad i = 1,2,3,\ldots$

m_i : plain-text bits.
k_i : key (key-stream) bits
c_i : cipher-text bits.

Example

- **Encryption:**
- 1001001 1000110 plaintext
- 1010110 0110001 key
- 0011111 1110110 ciphertext

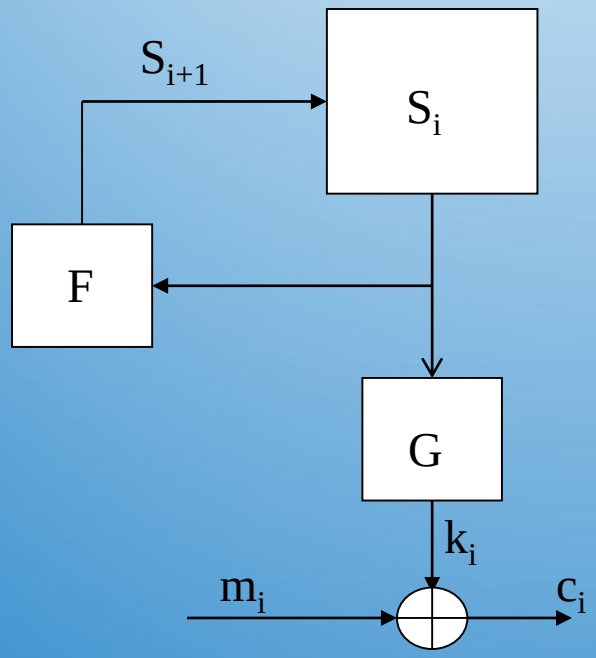
- **Decryption:**
- 0011111 1110110 ciphertext
- 1010110 0110001 key
- 1001001 1000110 plaintext

One-Time pad practical Problem

- Key-stream should be as long as plain-text
- Difficult in Key distribution & Management
- **Solution :**
  - Stream Ciphers
  - Key-stream is generated in pseudo-random fashion form Relatively short secret key

Stream Cipher Model

- **Output function** appears random



S_i : state of the cipher at time t = i.
F : state function.
G : output function.

Initial state, output and state functions are controlled by the secret key.

Random Numbers

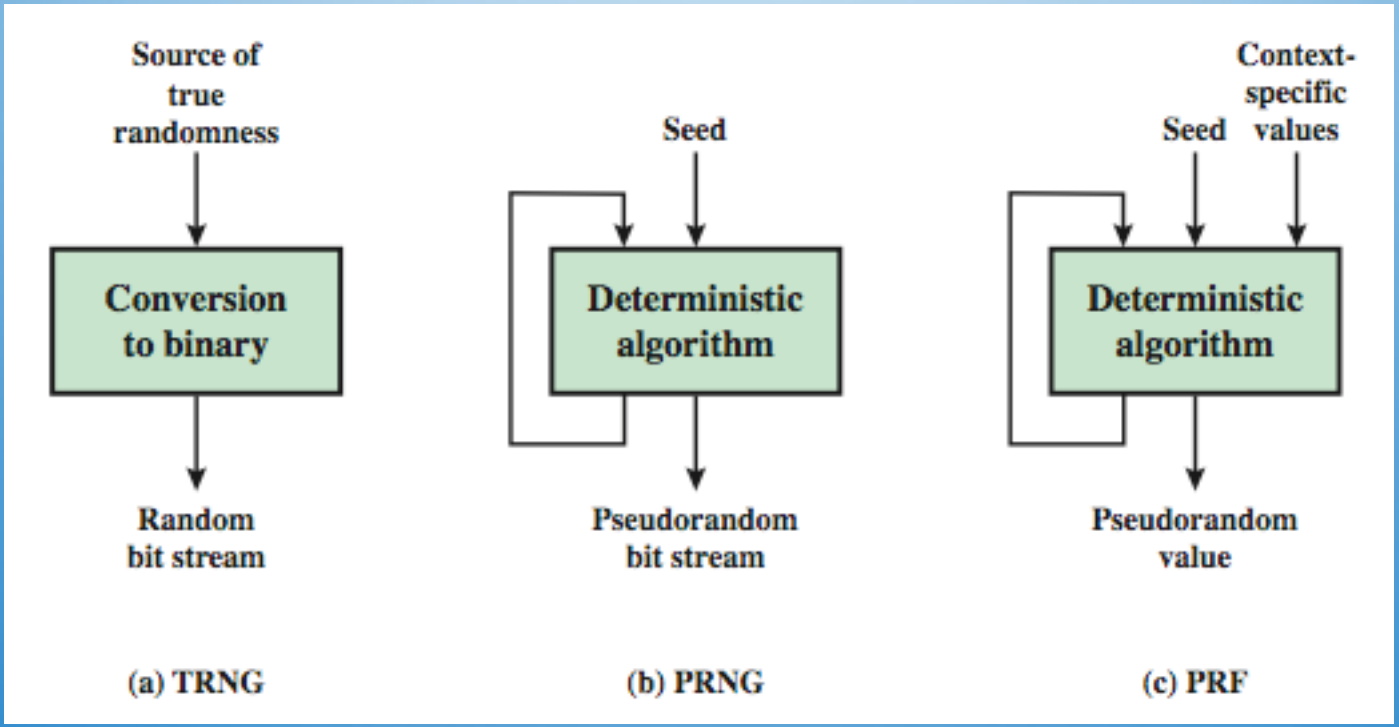
- Many uses of **random numbers** in cryptography
  - Nonce as Initialize Vector
  - Session keys
  - Public key generation
  - Keystream for a one-time pad
- In all cases its critical that these values be
  - statistically random, uniform distribution, independent
  - unpredictability of future values from previous values
- Care needed with generated random numbers

Pseudorandom Number Generators (PRNGs)

- Often use deterministic algorithmic techniques to create "random numbers"
  - although are not truly random
  - can pass many tests of "randomness"

- Known as "Pseudorandom Numbers"
- Created by "Pseudorandom Number Generators (PRNGs)"

Random & Pseudorandom Number Generators

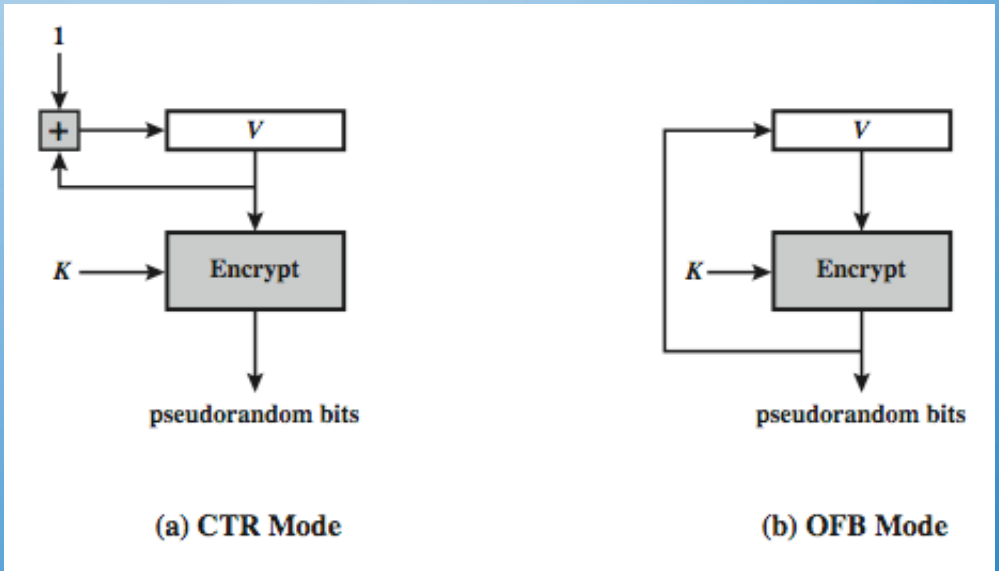


PRNG Requirements

- Randomness
  - uniformity, scalability, consistency
- Unpredictability
  - forward & backward Unpredictability
  - use same tests to check
- Characteristics of the seed
  - Secure
  - if known adversary can determine output
  - so must be random or pseudorandom number

Using Block Ciphers as PRNGs

- For cryptographic applications, can use a block cipher to generate random numbers
- Often for creating session keys from master key



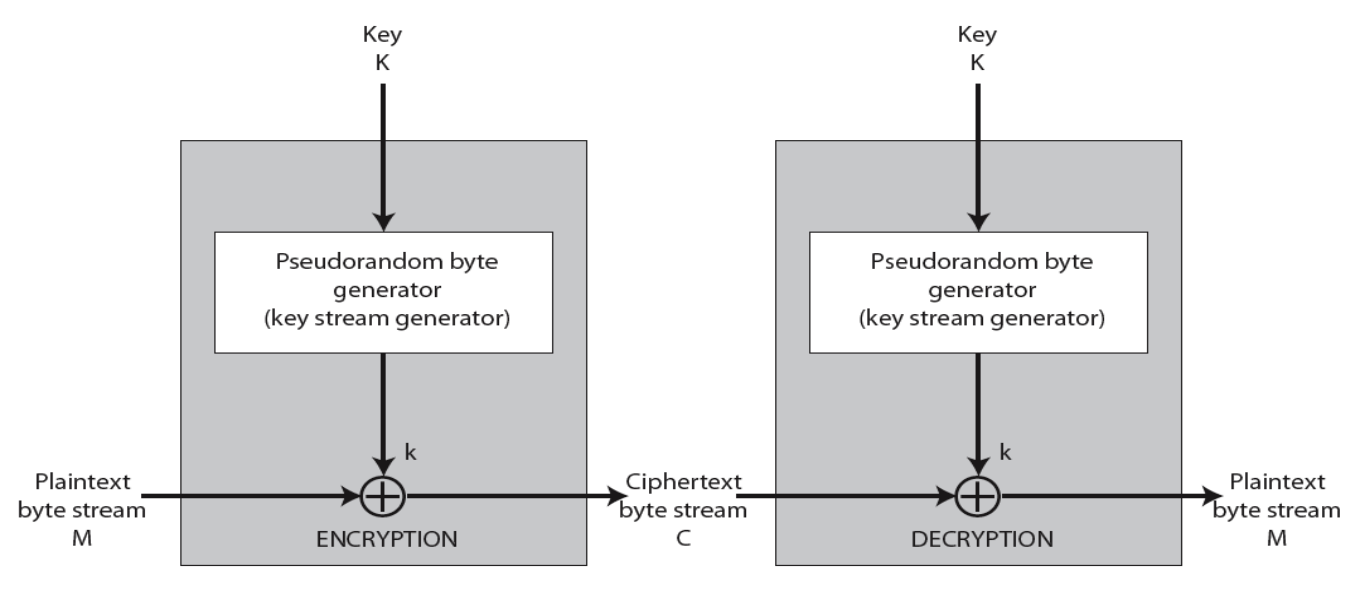
- **CTR**
  - $X_i = E_K[V_i]$
- **OFB**
  - $X_i = E_K[X_i\text{-}1]$

Stream Ciphers

- Generalization of **one-time pad**
- Stream cipher is initialized with short **key**
- Key is “stretched” into long **keystream**
  - have a pseudo random property
- Keystream is used like a one-time pad
  - XOR to encrypt or decrypt

Stream Cipher Structure

- Randomness of stream key completely destroys statistically properties in message
- Must **never reuse** stream key
  - otherwise can recover messages



Stream Cipher Properties

- Some design considerations are:
  - long period with no repetitions
  - statistically random
  - depends on large enough key
  - large linear complexity
- Properly designed, can be as secure as a block cipher with same size key
- Benefit : usually ***simpler*** & ***faster***

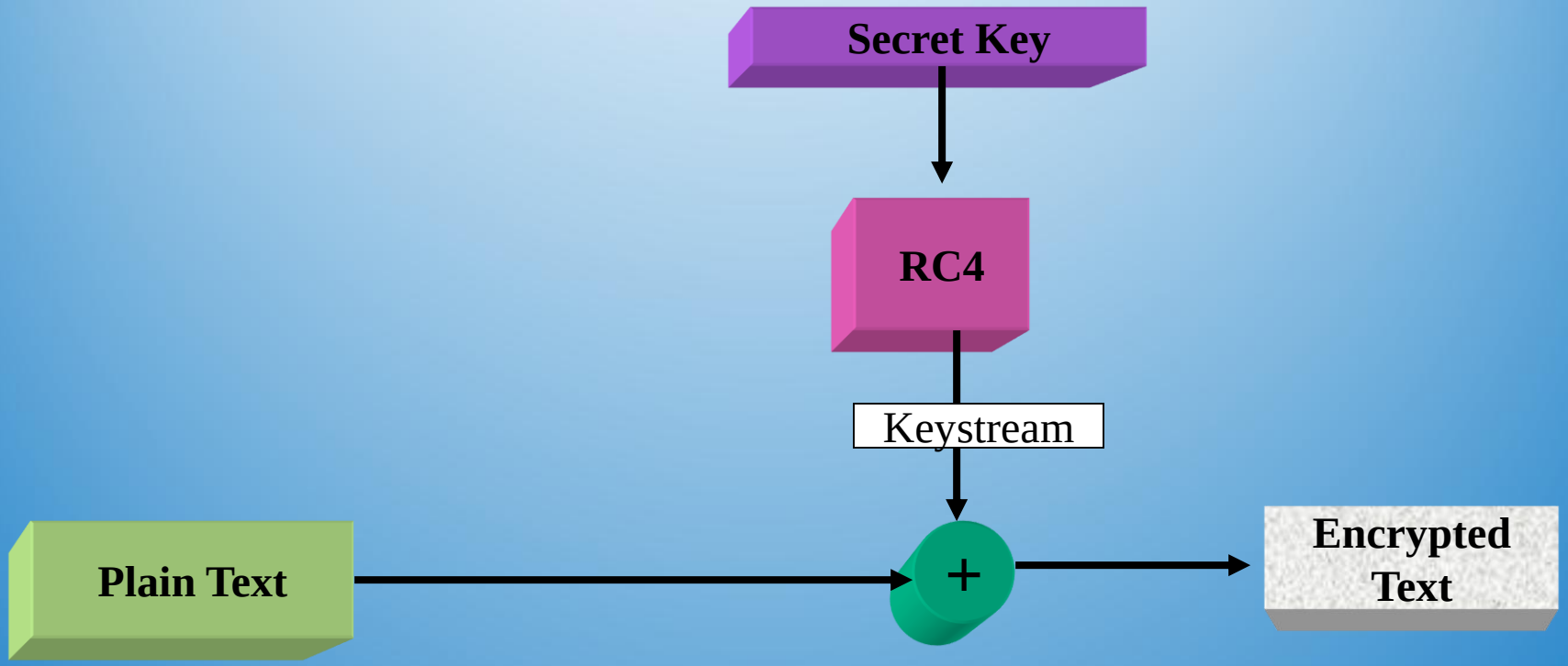
RC4 Basics

- A symmetric key encryption algorithm invented by Ron Rivest
  - A proprietary cipher owned by RSA, kept secret
  - Code released anonymously in Cyberpunks mailing list in 1994
  - Later posted sci.crypt newsgroup
- **Variable key size**, **byte-oriented** stream cipher
  - Normally uses 64 bit and 128 bit key sizes.
- Used in
  - SSL/TLS (Secure socket, transport layer security) between web browsers and servers,
  - IEEE 802.11 wirelss LAN std: WEP (Wired Equivalent Privacy), WPA (WiFi Protocol Access) protocol

RC4-based Usage

- WEP
- WPA default
- Bit Torrent Protocol Encryption
- Microsoft Point-to-Point Encryption
- SSL (optionally)
- SSH (optionally)
- Remote Desktop Protocol
- Kerberos (optionally)

RC4 Block Diagram



Cryptographically very strong and easy to implement

RC4 …Inside

- Consists of 2 parts:
  - Key Scheduling Algorithm (KSA)
  - Pseudo-Random Generation Algorithm (PRGA)

- KSA
  - Generate State array
- PRGA on the KSA
  - Generate keystream
  - XOR keystream with the data to generated encrypted stream

The KSA

- Use the secret key to initialize and **permutation** of state vector **S**, done in two steps
- Use 8-bit index pointers **i** and **j**

1

```
for i = 0 to 255 do
   S[i] = i;
   T[i] = K[i mod(|K|)];
```

[S], S is set equal to the values from 0 to 255
S[0]=0, S[1]=1,…, S[255]=255
[T], A temporary vector
[K], Array of bytes of secret key
|K| = Keylen, Length of (K)

2

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

- Use T to produce initial permutation of S
- The only operation on S is a swap;
  S still contains number from 0 to 255

After KSA, the input key and the temporary vector T will be no longer used

The PRGA

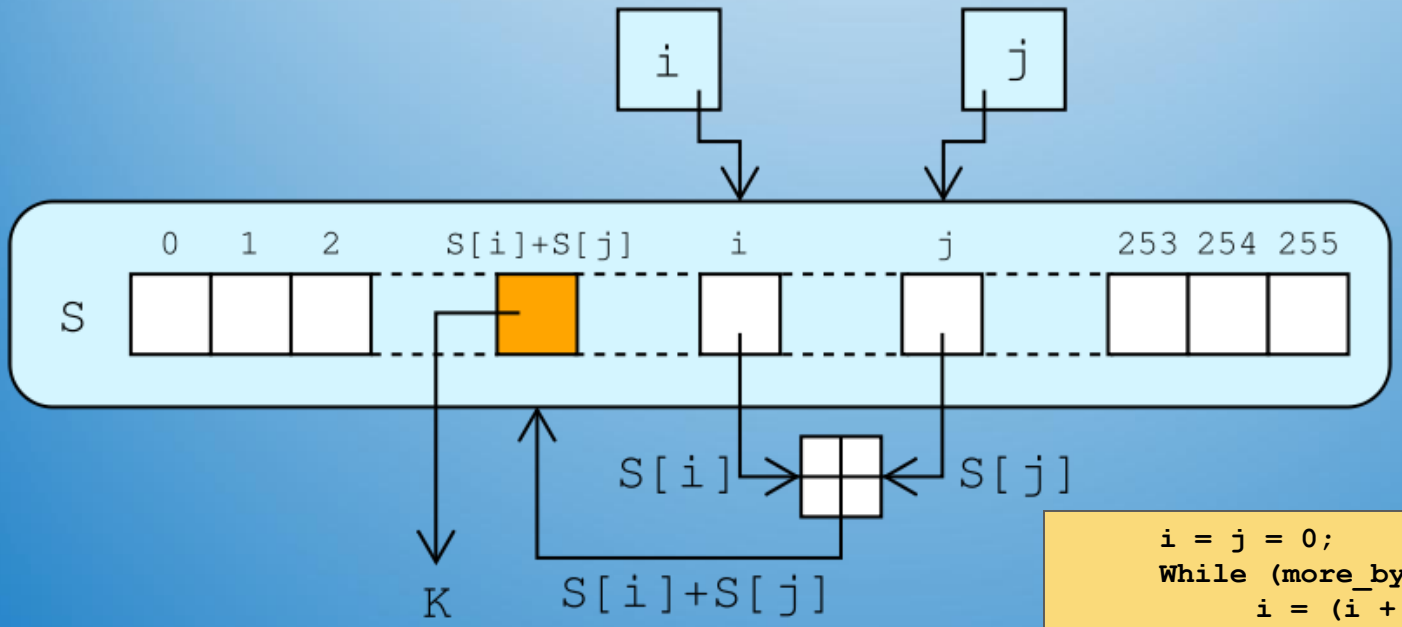
- Generate key stream *k* , one by one
- XOR S[k] with next byte of message to encrypt/decrypt

```
i = j = 0;
While (more_byte_to_encrypt)
    i = (i + 1) (mod 256);
    j = (j + S[i]) (mod 256);
    swap(S[i], S[j]);
    k = (S[i] + S[j]) (mod 256);
    C_i = M_i XOR S[k];
```

Sum of shuffled pair selects "stream key" value from permutation

RC4 Lookup Stage

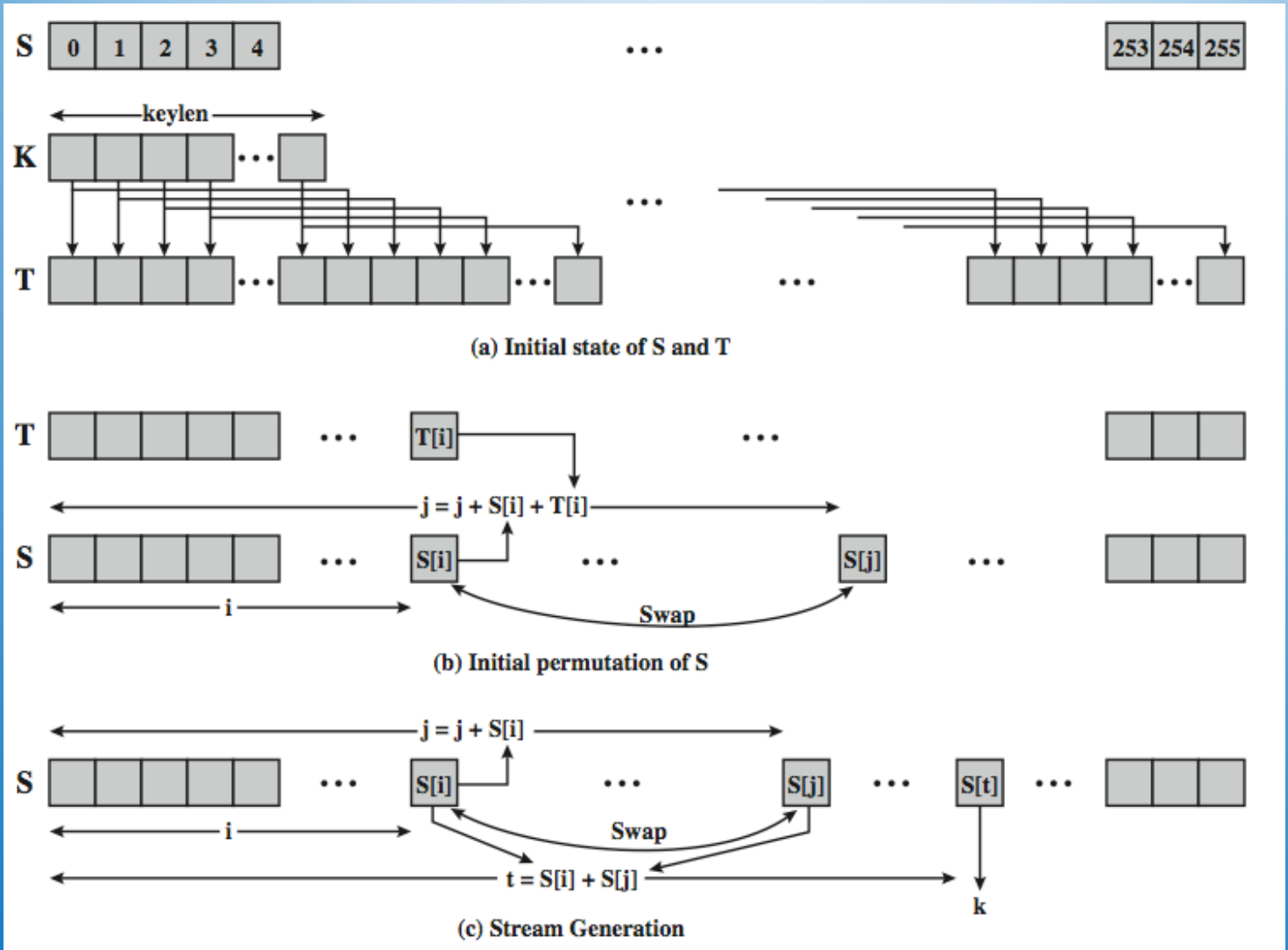
- The output byte is selected by looking up the values of S[i] and S[j], adding them together modulo 256, and then looking up the sum in S
- S [S[i] + S[j]] is used as a byte of the key stream, *K*



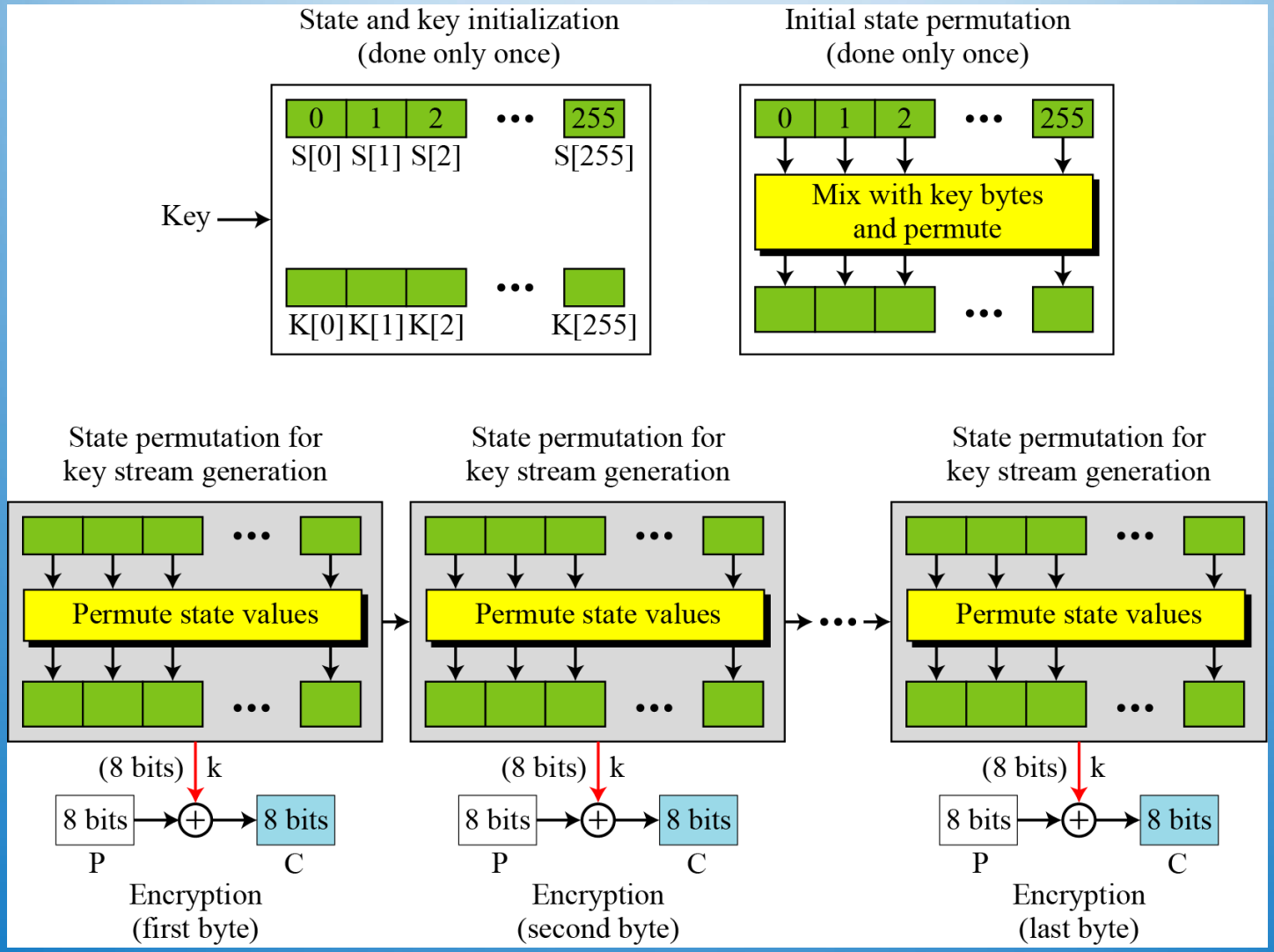
http://en.wikipedia.org/wiki/File:RC4.svg

```
i = j = 0;
While (more_byte_to_encrypt)
      i = (i + 1) (mod 256);
      j = (j + S[i]) (mod 256);
      swap(S[i], S[j]);
      k = (S[i] + S[j]) (mod 256);
      C_i = M_i XOR S[k];
```

Detailed Diagram



Overall Operation of RC4



Decryption using RC4

- Use the same secret key as during the encryption phase.
- Generate keystream by running the KSA and PRGA.
- XOR keystream with the encrypted text to generate the plain text.
- Logic is simple :

  (A xor B) xor B = A

  A = Plain Text or Data
  B = KeyStream

BLOCK CIPHER MODES OF OPERATION

What is block cipher?

- In cryptography, a block cipher is a deterministic algorithm operating on fixed-length groups of bits, called blocks, with an unvarying transformation that is specified by a symmetric key.
- A block cipher algorithm is a basic building block for providing data security.
- To apply a block cipher in a variety of applications, five "modes of operations" are defined by NIST.

Introduction to block cipher modes of operation

- NIST : National Institute of Standards and Technology
- That five modes of operations are :

1. Electronic code book
2. Cipher chaining block
3. Cipher feedback mode
4. Output feedback mode
5. Counter mode

When we use block cipher modes of operation?

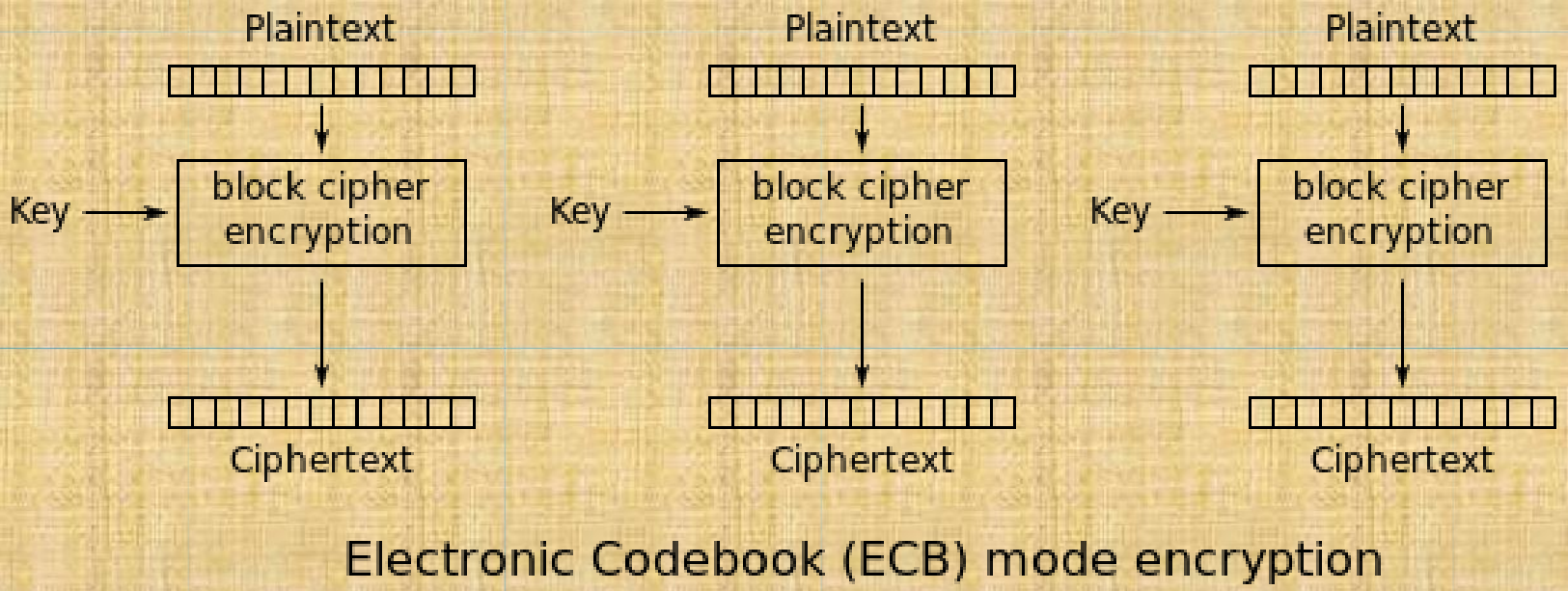
- Block cipher only allow to encrypt entire blocks.
- What if our message is longer/shorter than the block size?
- When message is longer/shorter than the block size , we use modes of operations.
- Algorithms that exploit a block cipher to provide a service (e.g. confidentiality).

Electronic codebook

- ECB is the simplest mode of operation.
- The plain text is divided into N blocks.
- The block size is n bits.
- If the plaintext size is not multiple of the block size , the text is padded to make the last block the same size other blocks.
- Same key is used to encrypt and decrypt each block.

Electronic codebook encryption process

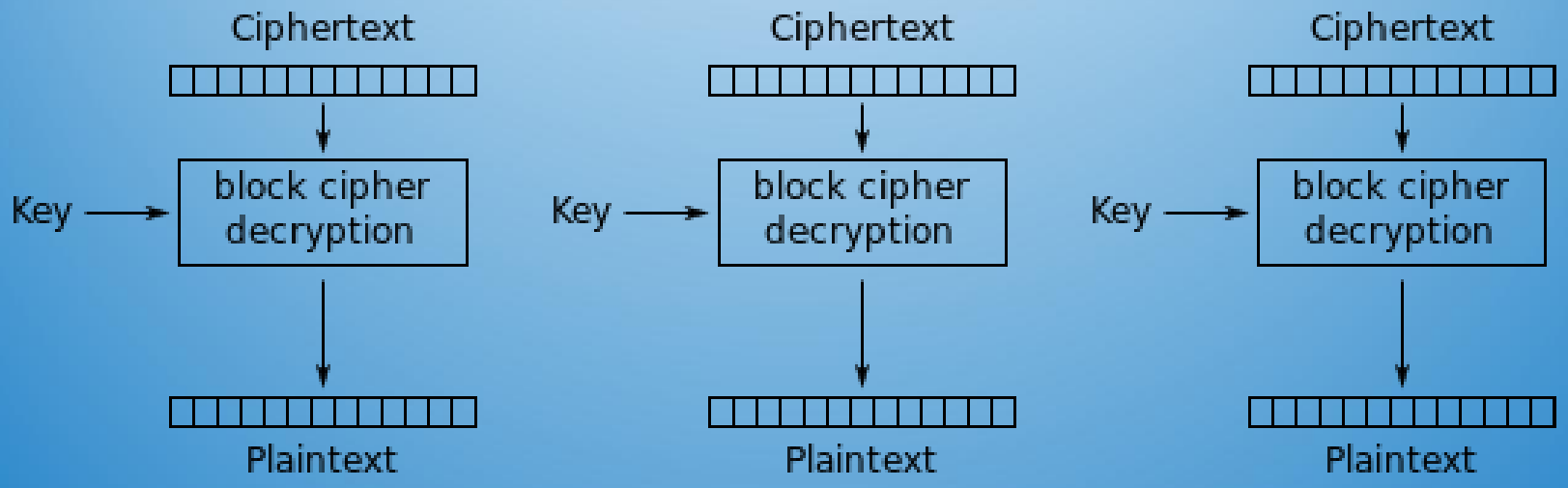
Cipher texti = encryption with key (plain texti)



Electronic Codebook (ECB) mode encryption

Electronic codebook Decryption process

Plain text$_i$ = Decryption with key (cipher text$_i$)



Electronic Codebook (ECB) mode decryption

Electronic codebook encryption/decryption Security issues

- Patterns at the block level are preserved.
- For example equal blocks in the plain text become equal block in the cipher text.
- If any person finds out the cipher text block 1,5 and 10 are the same ,that person knows that plaintext blocks 1, 5 and 10 are the same.
- This is a leak in security.

What is initialization vector?

- An initialization vector (IV) or starting variable is a block of bits that is used by several modes to randomize the encryption and hence to produce distinct cipher texts even if the same plain text is encrypted multiple times, without the need for a slower re-keying process.
- An initialization vector has different security requirements than a key, so the IV usually does not need to be secret
- However, in most cases, it is important that an initialization vector is never reused under the same key.

What is initialization vector? (continue…)

- For CBC and CFB, reusing an IV leaks some information about the first block of plaintext, and about any common prefix shared by the two messages.
- For OFB and CTR, reusing an IV completely destroys security. This can be seen because both modes effectively create a bit stream that is XORed with the plaintext, and this bit stream is dependent on the password and IV only. Reusing a bit stream destroys security.
- In CBC mode, the IV must, in addition, be unpredictable at encryption time; in particular, the (previously) common practice of re-using the last cipher text block of a message as the IV for the next message is insecure.

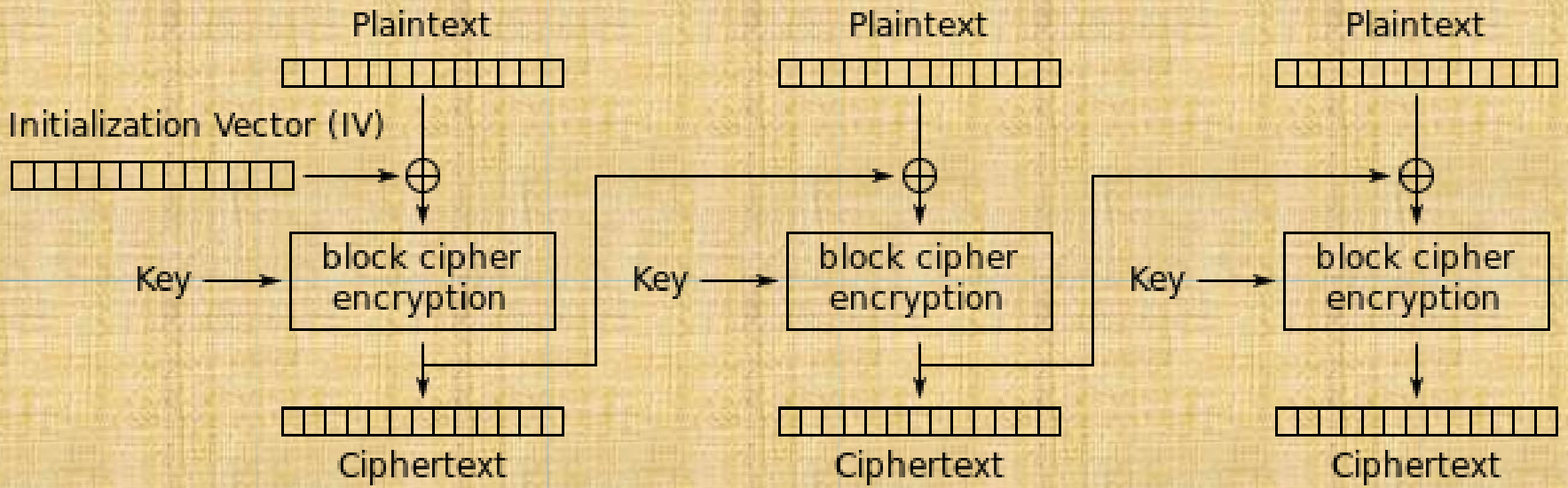
Cipher block chaining mode

- IBM invented the Cipher Block Chaining (CBC) mode of operation in 1976.
- In CBC mode, each block of plaintext is XORed with the previous cipher text block before being encrypted.
- This way, each cipher text block depends on all plaintext blocks processed up to that point. To make each message unique, an initialization vector must be used in the first block.

Cipher block chaining mode encryption

IV = initialization vector

Cipher text$_i$ = encryption with key (plain text XOR cipher text $_{i-1}$)

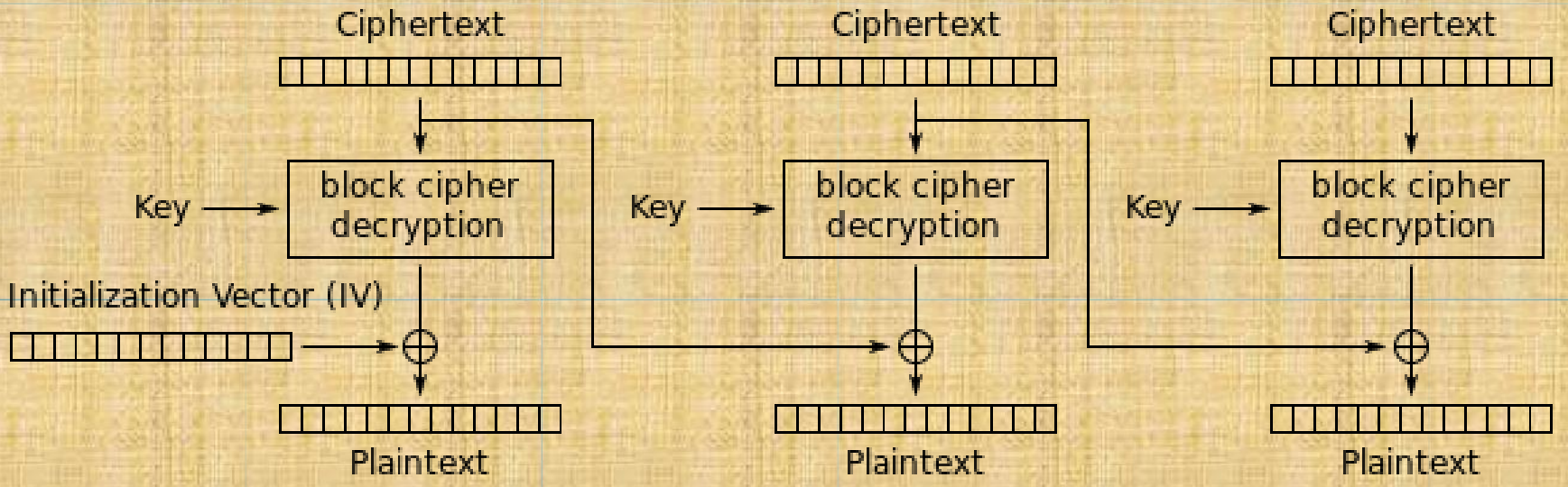


Cipher Block Chaining (CBC) mode encryption

Cipher block chaining mode Decryption

IV = initialization vector

plain text$_i$ = Decryption with key (cipher text XOR cipher text $_{i-1}$)



Cipher Block Chaining (CBC) mode decryption

Cipher block chaining mode Security issues

- The patterns at the block level are not preserved.
- In CBC mode, equal plain text block belonging to the same message are enciphered into different cipher text block.
- However ,if two message are equal ,their encipherment is the same if they use the same IV.
- As a matter of fact ,if the first M blocks in two different message are equal , they are enciphered into equal blocks unless different IVs are used.
- For this reason , some people recommended the use of time-stamp as an IV.
- Any person can add some cipher text blocks to the end of the cipher text stream.

Cipher feedback mode

- ECB and CBC modes encrypt and decrypt blocks of the message.
- Block size n is predetermine by the underlying cipher ; for example , for DES n = 64

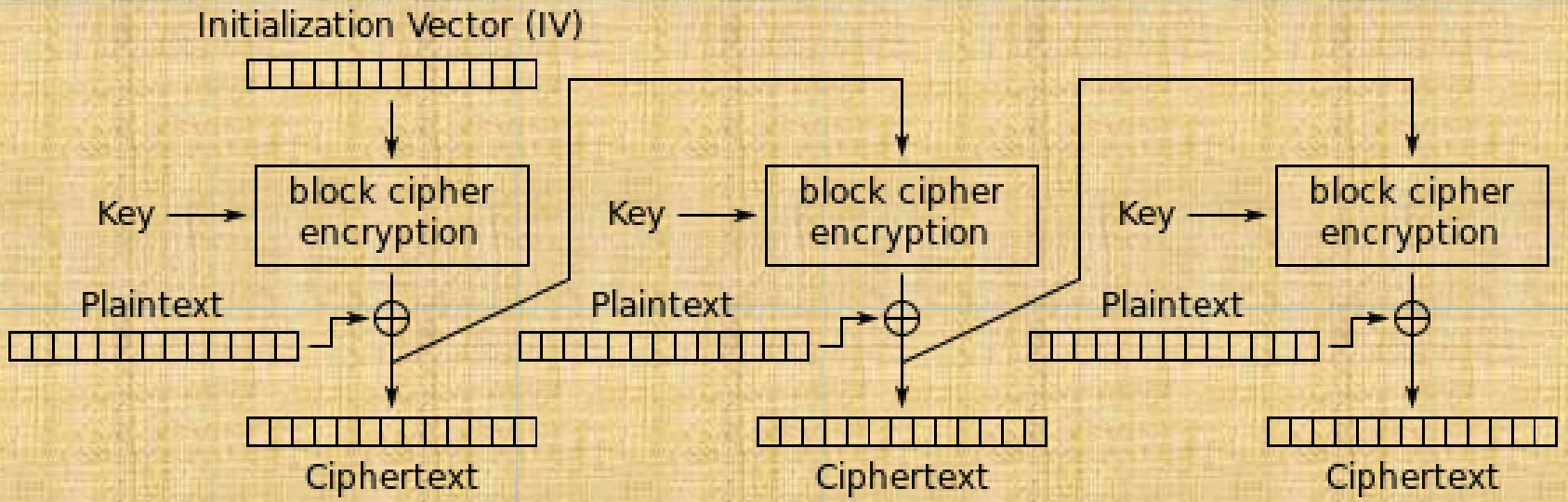
for AES n =128

- In some situations, we need use DES or AES as secure cipher , but the plain text or cipher text block size are to be smaller.
- For example , to encrypt and decrypt 8-bit characters , you would not want to use one of the traditional cipher like Caesar cipher.
- The solution is to use DES or AES in cipher feedback mode

Cipher feedback mode encryption

IV = initialization vector

Cipher text$_i$ = Encryption with key (Cipher text $_{i-1}$) XOR plain text

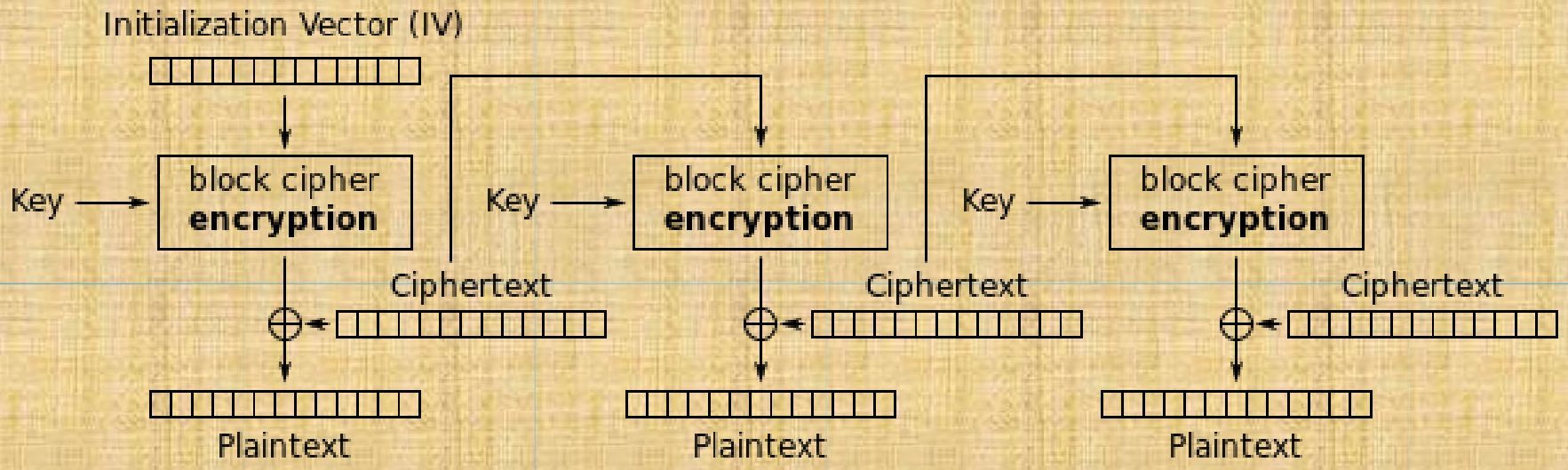


Cipher Feedback (CFB) mode encryption

Cipher feedback mode Decryption

IV = initialization vector

Plain $text_i$ = Encryption with key (Cipher $text_{i-1}$) XOR cipher text



Cipher Feedback (CFB) mode decryption

Cipher feedback mode Security issues

- Just like CBC , patterns at the block level are not preserved.
- More than one message can be encrypted with the same key , but the value of the IV should be changed for each message.
- This means that sender needs to use a different IV each time sender sends a message.
- Attacker can add some cipher text block to the end of the cipher text stream

Cipher output feedback mode

- Output feedback mode is very similar to CFB mode , with one difference: each bit in the cipher text is independent of the previous bit or bits.
- This avoids error propagation.
- If an error occur in transmission , it does not affect the bits that follow.
- Note that , like cipher feedback mode , both the sender and the receiver use the encryption algorithm.

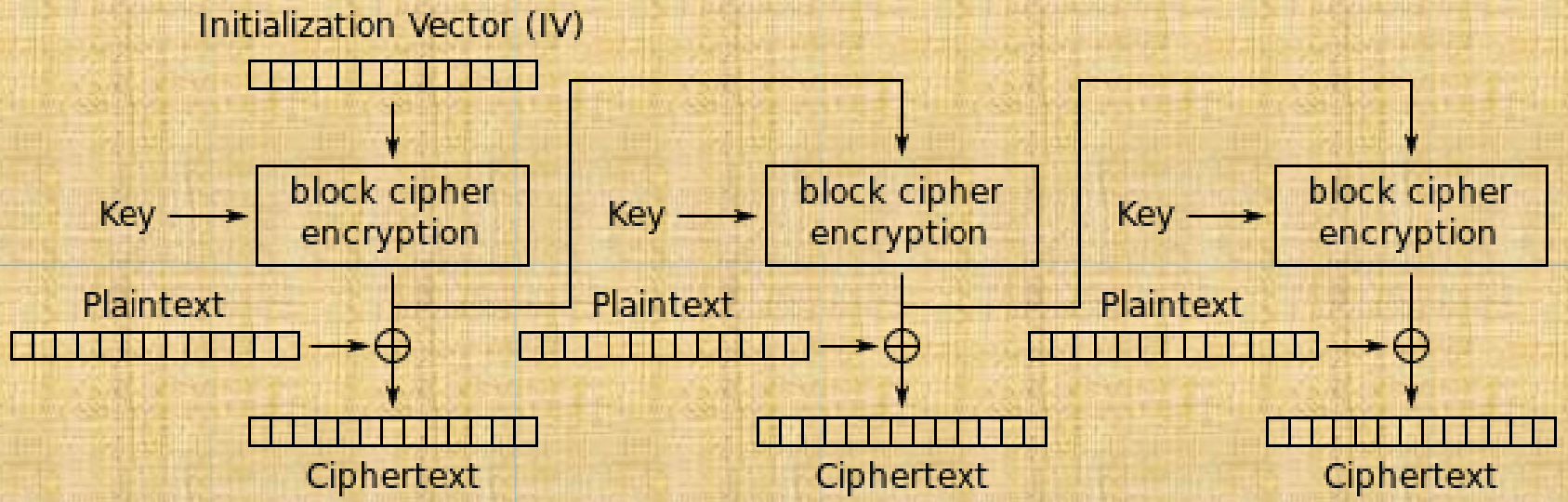
Cipher output feedback mode encryption

IV = initialization vector cipher text$_i$

=

plain text$_i$ XOR Encryption

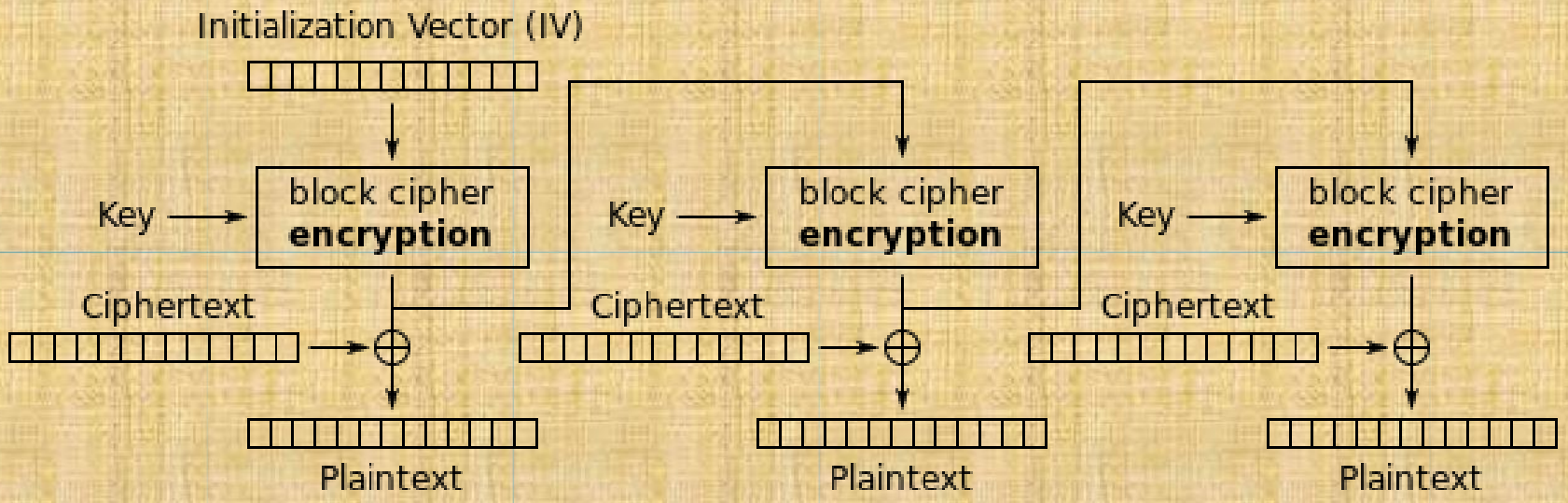
(k , [cipher text i-1 XOR plain text i-1])



Output Feedback (OFB) mode encryption

Cipher output feedback mode Decryption

IV = initialization vector Plain text$_i$ =
cipher texti XOR Encryption
(k , [cipher text i-1 XOR plain text $_{i-1}$])



Output Feedback (OFB) mode decryption

Cipher output feedback mode Security issues

- Just like CBC , patterns at the block level are not preserved.
- Any change in the cipher text affects the plain text encrypted at the receiver side.

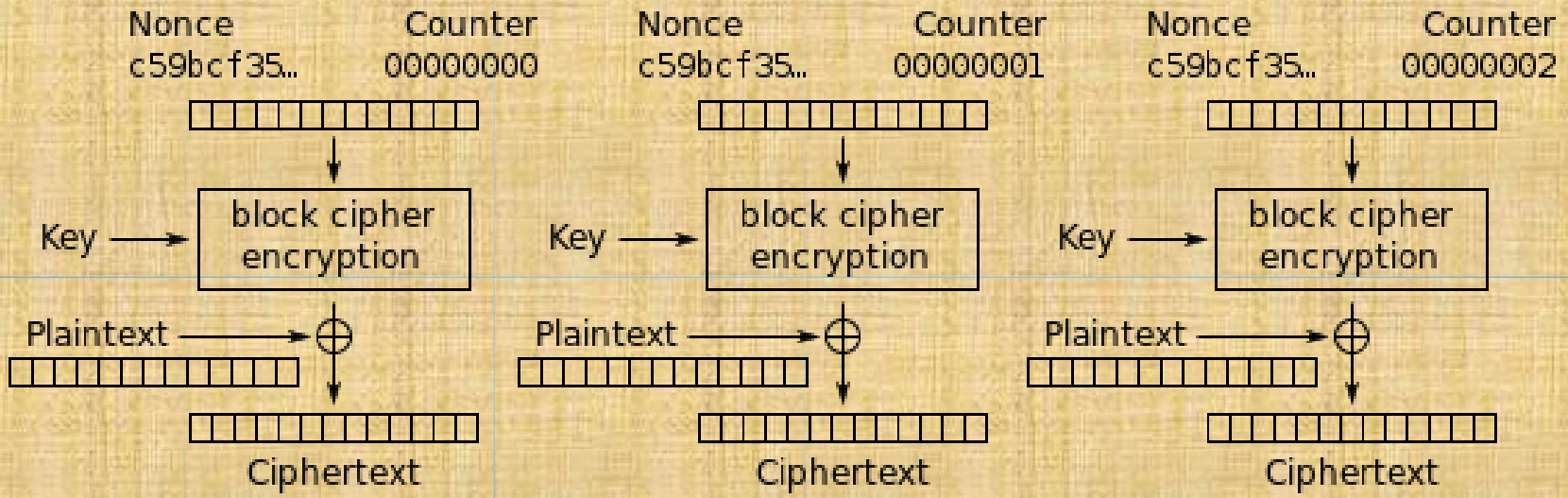
Counter

- In the counter mode , there is no feedback.
- The pseudo randomness in the key streams achieved using a counter.
- An n bit counter is initialized to a predetermined value(IV) and incremented based on a predefined rule(mod 2^n)
- To provide a better randomness , the increment value can depend on the block numbers to be incremented.
- The plain text and cipher block text block have same block size as the underlying cipher.
- Both encryption and decryption can be performed fully in parallel on multiple blocks .
- Provides true random access to cipher text blocks

Counter Encryption process

Nonce = IV

Cipher $text_i$ = Plain $text_i$ XOR Encryption with key(counter)

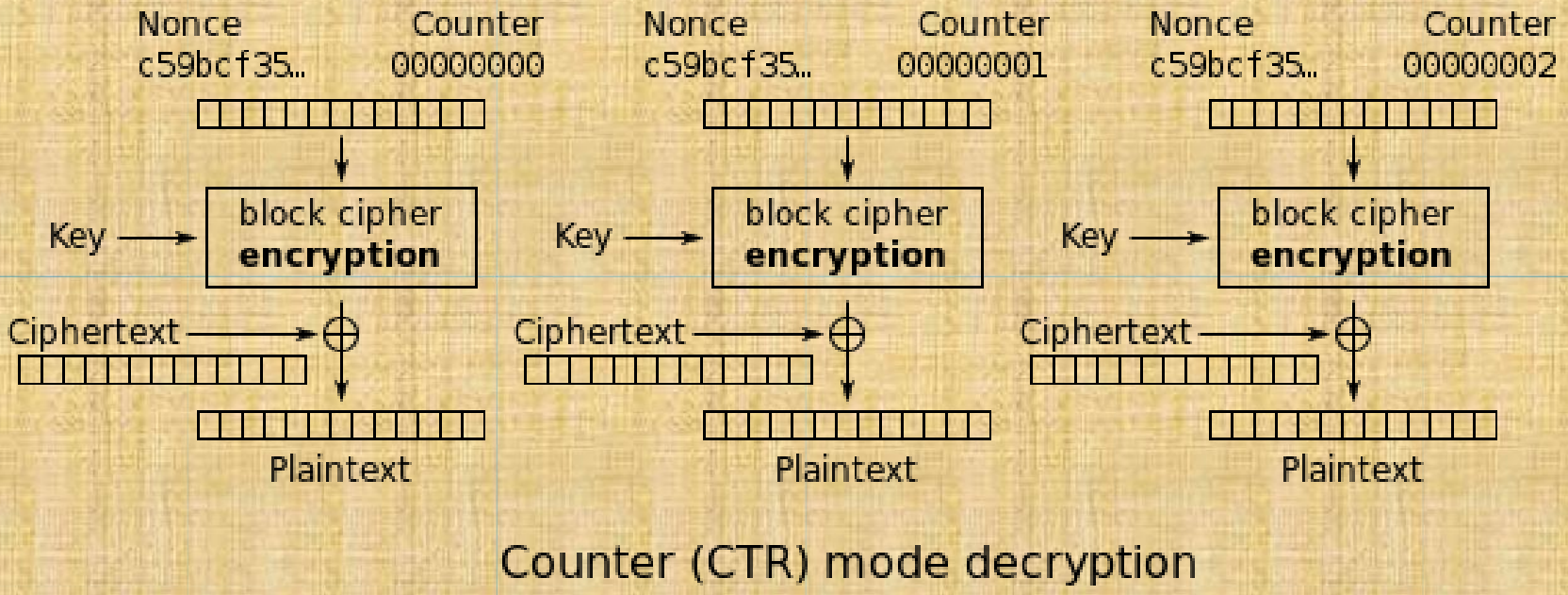


Counter (CTR) mode encryption

Counter Decryption process

Nonce = IV

Plain text$_i$ = Cipher text$_i$ XOR Encryption with key(counter)

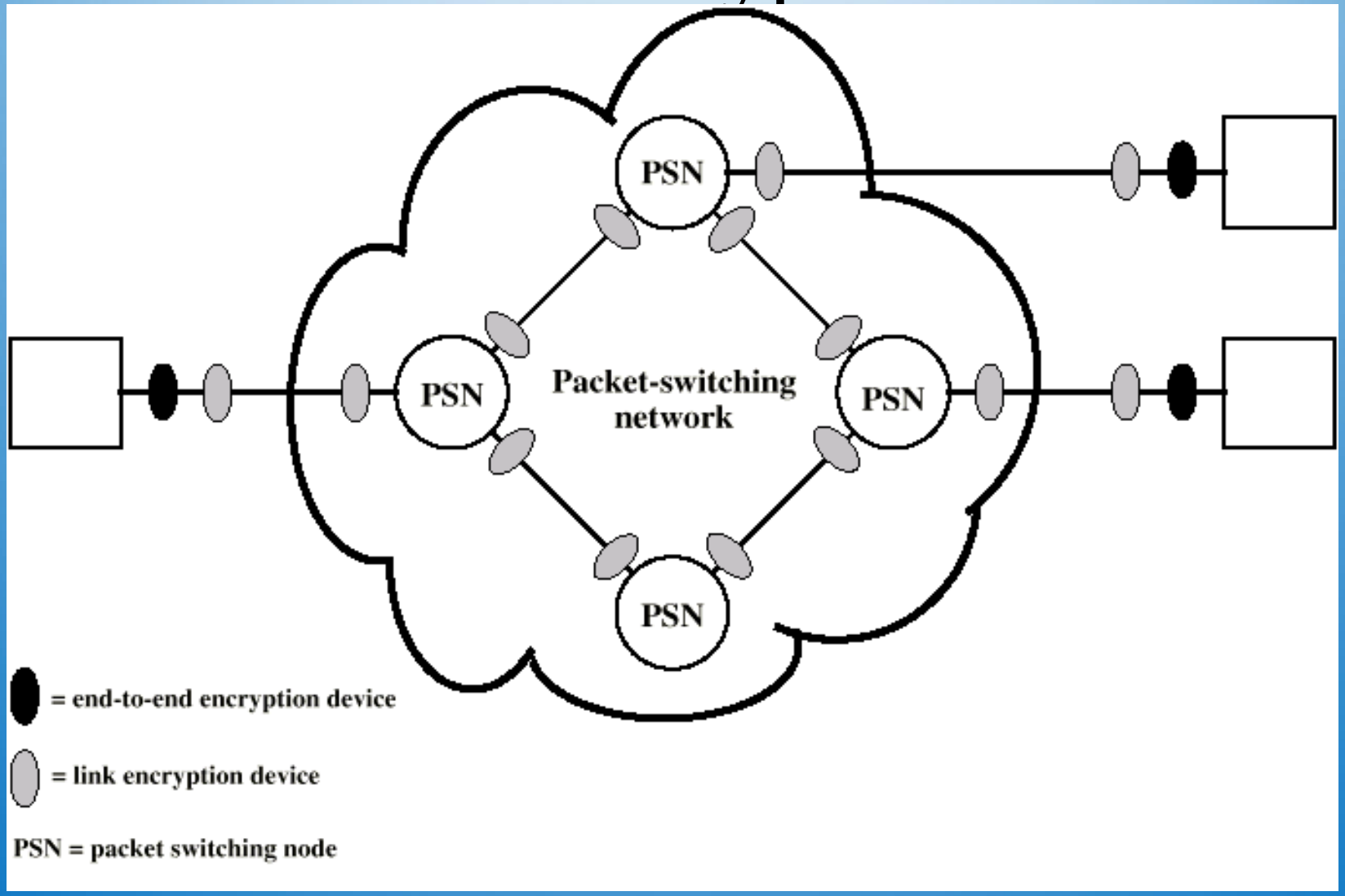


Counter (CTR) mode decryption

Location of Encryption Devices

- The most powerful, and most common, approach to countering the threats to network security is encryption.
- In using Encryption, we need to decide what to encrypt and where the encryption gear should be located. There are two fundamental alternatives:
  - Link
  - End to End

Location of Encryption Devices



Link Encryption

- Each communication link equipped at both ends
- All traffic secure
- High level of security
- Requires lots of encryption devices
- Message must be decrypted at each switch to read address (virtual circuit number)
- Security vulnerable at switches
  - Particularly on public switched network

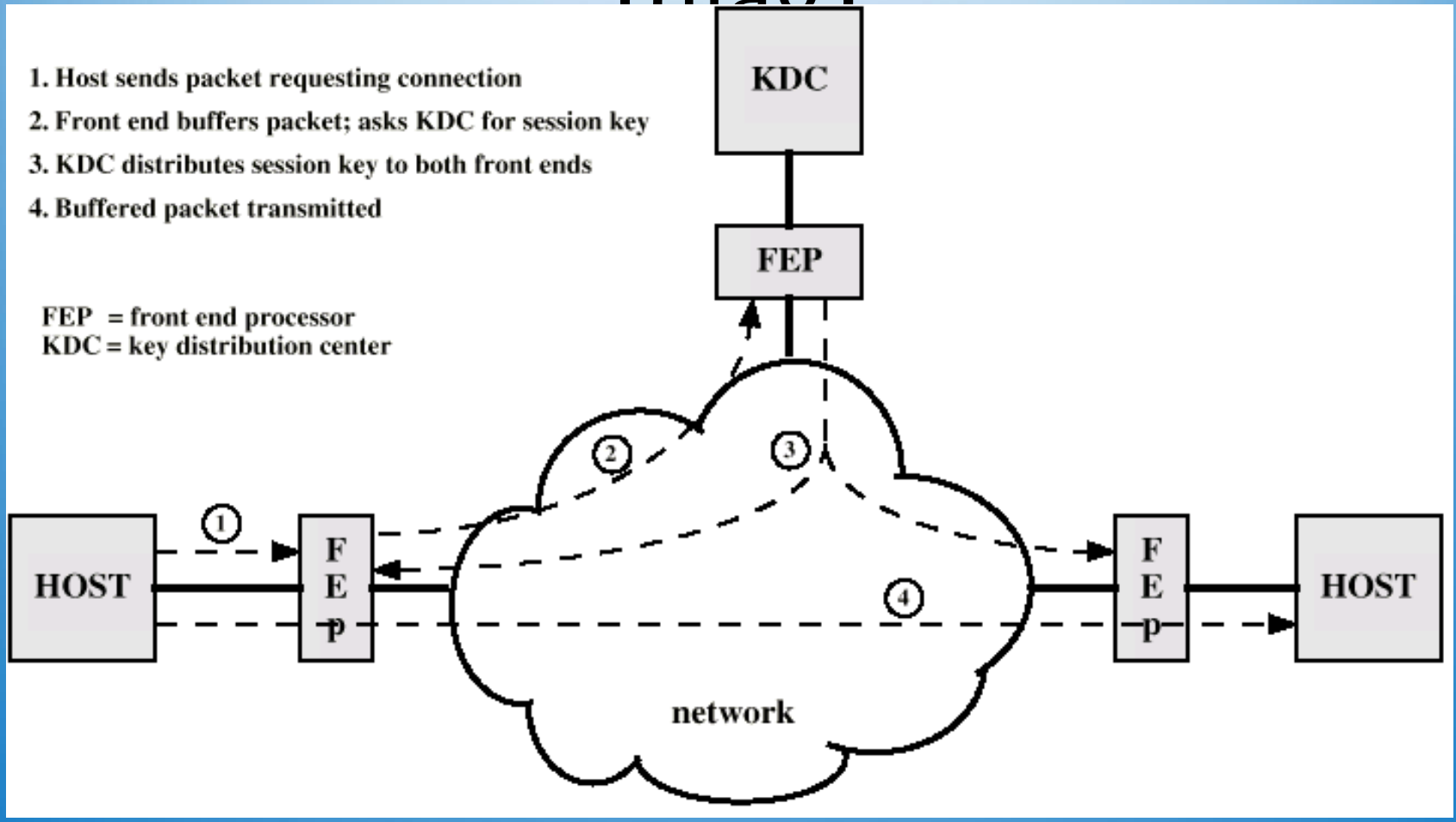
End to End Encryption

- Encryption done at ends of system
- Data in encrypted form crosses network unaltered
- Destination shares key with source to decrypt
- Host can only encrypt user data
  - Otherwise switching nodes could not read header or route packet
- Traffic pattern not secure

Key Distribution

- Key selected by A and delivered to B
- Third party selects key and delivers to A and B
- Use old key to encrypt and transmit new key from A to B
- Use old key to transmit new key from third party to A and B

Automatic Key Distribution (diag)



Automatic Key Distribution

- Session Key
  - Used for duration of one logical connection
  - Destroyed at end of session
  - Used for user data
- Permanent key
  - Used for distribution of keys
- Key distribution center
  - Determines which systems may communicate
  - Provides one session key for that connection

Approaches to Message Authentication

- Encryption protects against passive attack (eavesdropping). A different requirement is to protect against active attack (falsification of data and transactions).
- Protection against such attacks is known as message authentication.
- A message, file, document, or other collection of data is said to be authentic when it is genuine and comes from its alleged source.
- Message authentication is a procedure that allows communicating parties to verify that received messages are authentic.
- The two important aspects are to verify that the contents of the message
- have not been altered and that the source is authentic.
- We may also wish to verify a message's timeliness (it has not been artificially delayed and replayed) and sequence relative to other messages flowing between two parties.
- All of these concerns come under the category of data integrity as described in Chapter 1.

Authentication Using Conventional Encryption

- It would seem possible to perform authentication simply by the use of symmetric encryption. If we assume that only the sender and receiver share a key then only the genuine sender would be able to encrypt a message successfully for the other participant, provided the receiver can recognize a valid message.
- Furthermore, if the message includes an error-detection code and a sequence number, the receiver is assured that no alterations have been made and that sequencing is proper.
- If the message also includes a timestamp, the receiver is assured that the message has not been delayed beyond that normally expected for network transit.
- In fact, symmetric encryption alone is not a suitable tool for data authentication.

- To give one simple example, in the ECB mode of encryption, if an attacker reorders the blocks of ciphertext, then each block will still decrypt successfully.
- However, the reordering may alter the meaning of the overall data sequence.
- Although sequence numbers may be used at some level (e.g., each IP packet), it is
- typically not the case that a separate sequence number will be associated with each b-bit block of plaintext.
- Thus, block reordering is a threat.

Message Authentication without Message Encryption

- In this section, we examine several approaches to message authentication that do not rely on encryption.
- In all of these approaches, an authentication tag is generated and appended to each message for transmission.
- The message itself is not encrypted and can be read at the destination independent of the authentication function at the destination.
- Because the approaches discussed in this section do not encrypt the message, message confidentiality is not provided.
- As was mentioned, message encryption by itself does not provide a secure form of authentication. However, it is possible to combine authentication and confidentiality in a single algorithm by encrypting a message plus its authentication tag.
- Typically, however, message authentication is provided as a separate function from message encryption. [DAVI89] suggests three situations in which message authentication without confidentiality is preferable:

1) There are a number of applications in which the same message is broadcast to a number of destinations.

- Two examples are notification to users that the network is now unavailable and an alarm signal in a control center.
- It is cheaper and more reliable to have only one destination responsible for monitoring authenticity.Thus, the message must be broadcast in plaintext with an associated message authentication tag. The responsible system performs authentication.
- If a violation occurs, the other destination systems are alerted by a general alarm.

2) Another possible scenario is an exchange in which one side has a heavy load and cannot afford the time to decrypt all incoming messages.

- Authentication is carried out on a selective basis with messages being chosen at random for checking.

3) Authentication of a computer program in plaintext is an attractive service.The computer program can be executed without having to decrypt it every time,

- which would be wasteful of processor resources.
- However, if a message authentication tag were attached to the program, it could be checked whenever assurance is required of the

Thus, there is a place for both authentication and encryption in meeting security requirements.

Message Authentication Code

- One authentication technique involves the use of a secret key to generate a small block of data, known as a **message authentication code** (MAC), that is appended to the message.
- This technique assumes that two communicating parties, say A and B, share a common secret key *KAB. When A has a* message to send to B, it calculates the message authentication code as a function of the message and the key: MAC*M F(KAB,M).The message plus code are transmitted to* the intended recipient.
- The recipient performs the same calculation on the received message, using the same secret key, to generate a new message authentication code.
- The received code is compared to the calculated code (Figure 3.1). If we assume that only the receiver and the sender know the identity of the secret key, and if the received code matches the calculated code, then the following statements apply:

- The receiver is assured that the message has not been altered. If an attacker alters the message but does not alter the code, then the receiver's calculation of the code will differ from the received code.
- Because the attacker is assumed not to know the secret key, the attacker cannot alter the code to correspond to the alterations in the message.

2. The receiver is assured that the message is from the alleged sender. Because no one

else knows the secret key, no one else could prepare a message with a proper code.

3. If the message includes a sequence number (such as is used with HDLC and

TCP), then the receiver can be assured of the proper sequence, because an attacker cannot successfully alter the sequence number.

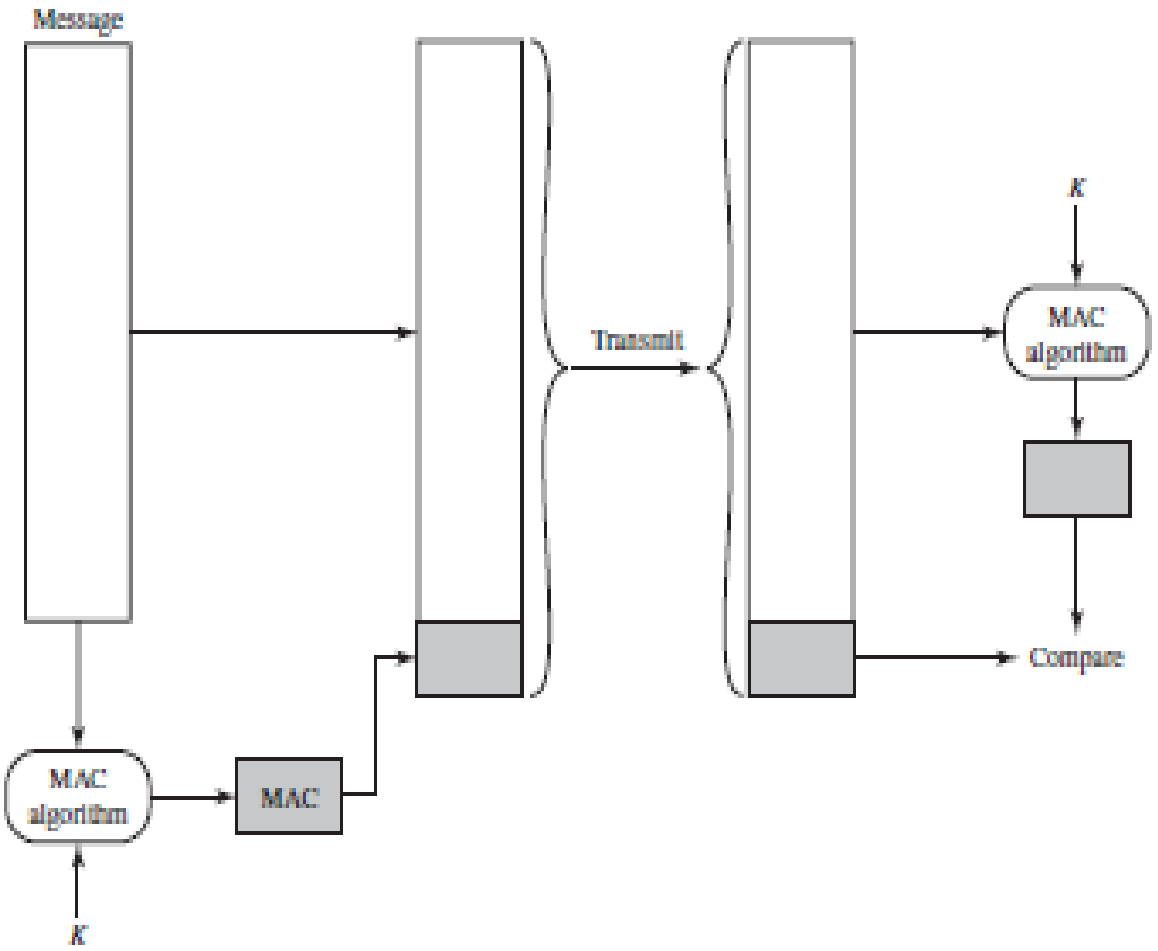


Figure 3.1 Message Authentication Using a Message Authentication Code (MAC)

OUTLINE

- Approaches to Message Authentication
- Secure Hash Functions and HMAC

Authentication

- Requirements - must be able to verify that:
  1. Message came from apparent source or author,
  2. Contents have not been altered,
  3. Sometimes, it was sent at a certain time or sequence.
- Protection against active attack (falsification of data and transactions)

Approaches to Message Authentication

- Authentication Using Conventional Encryption
  - Only the sender and receiver should share a key
- Message Authentication without Message Encryption
  - An authentication tag is generated and appended to each message
- Message Authentication Code
  - Calculate the MAC as a function of the message and the key. MAC = F(K, M)

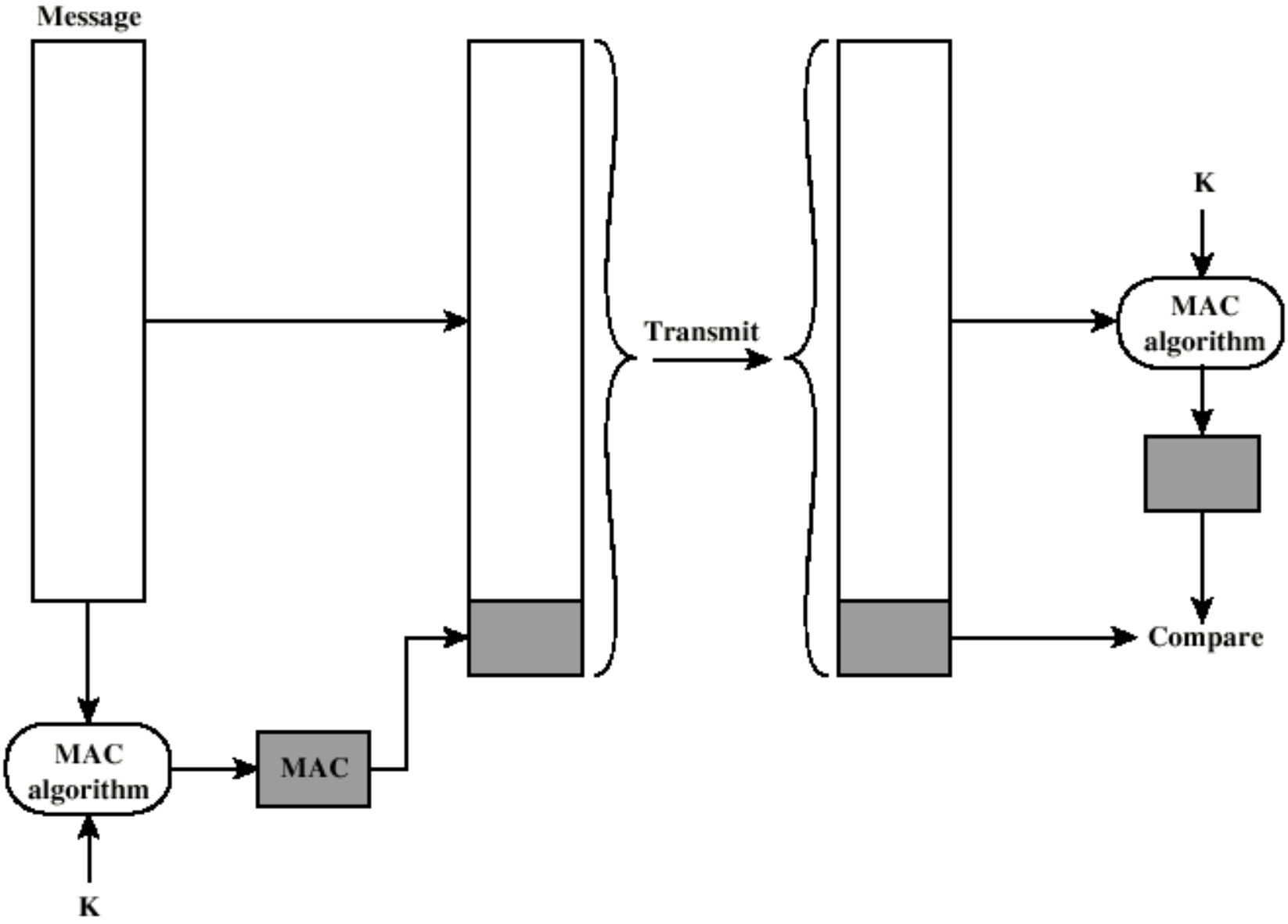
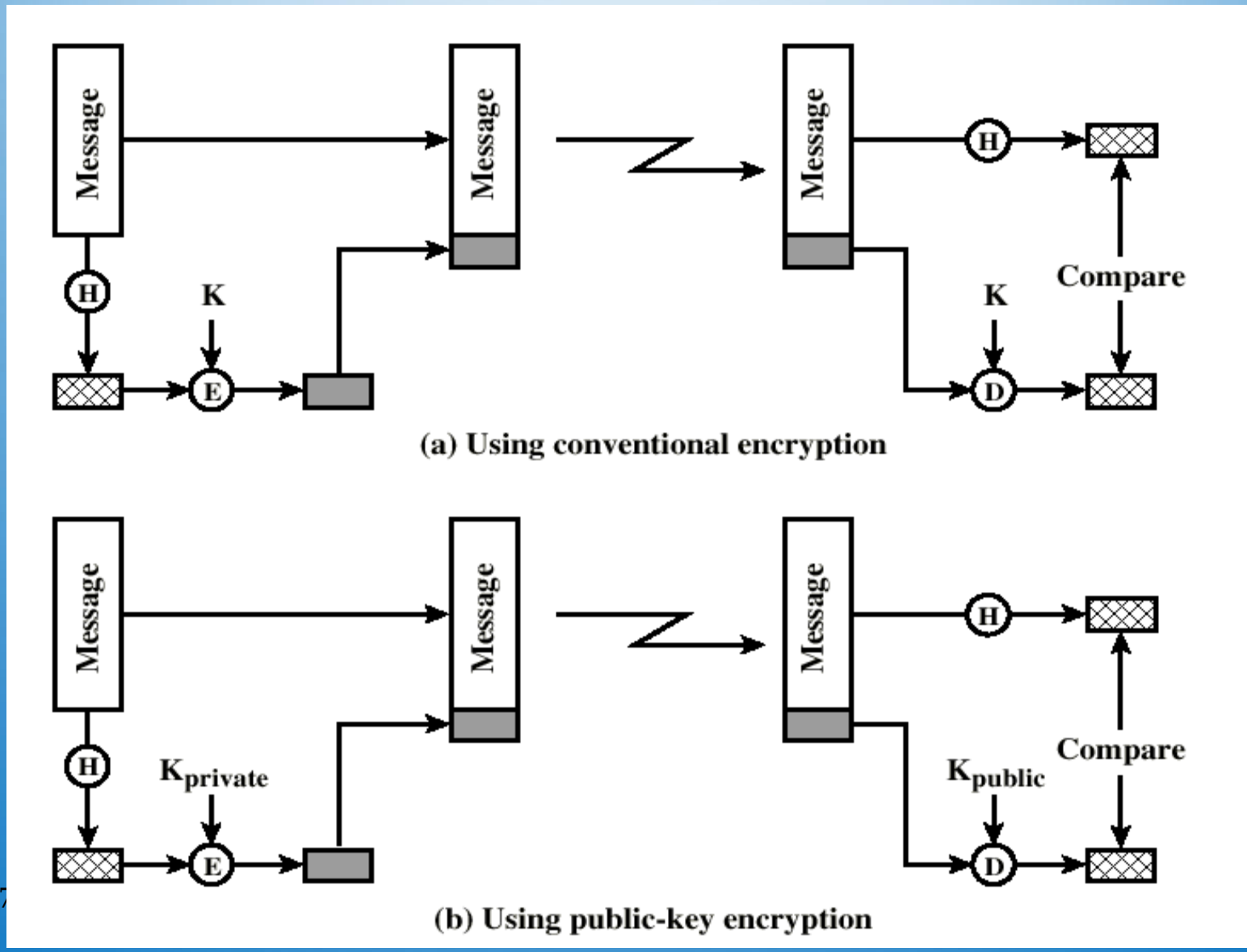


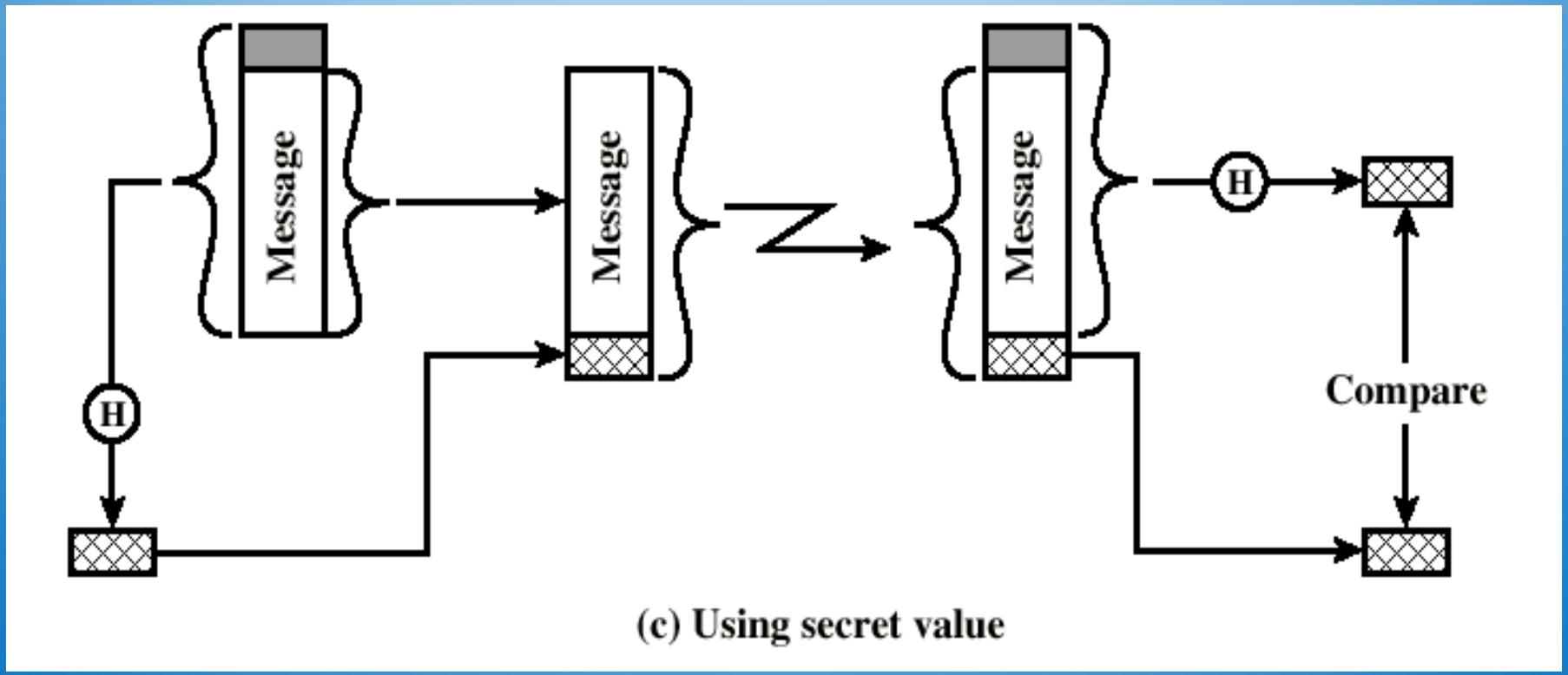
Figure 3.1 Message Authentication Using a Message Authentication Code (MAC)

One-way HASH function



One-way HASH function

- Secret value is added before the hash and removed before transmission.



(c) Using secret value

Secure HASH Functions

- Purpose of the HASH function is to produce a ”fingerprint.
- Properties of a HASH function H :
  1. H can be applied to a block of data at any size
  2. H produces a fixed length output
  3. H(x) is easy to compute for any given x.
  4. For any given block x, it is computationally infeasible to find x such that H(x) = h
  5. For any given block x, it is computationally infeasible to find $y \neq x$ with H(y) = H(x).
  6. It is computationally infeasible to find any pair (x, y) such that H(x) = H(y)

Security of Hash Functions

- As with symmetric encryption, there are two approaches to attacking a secure hash function: cryptanalysis and brute-force attack.
- As with symmetric encryption algorithms, cryptanalysis of a hash function involves exploiting logical weaknesses in the algorithm.
- The strength of a hash function against brute-force attacks depends solely on the length of the hash code produced by the algorithm.
- For a hash code of length *n*, the level of effort required is proportional to the following:

- The first three properties are requirements for the practical application of a hash function to message authentication.
- The fourth property, preimage resistant, is the “one-way” property: It is easy to generate a code given a message, but virtually impossible to generate a message given a code.
- This property is important if the authentication technique involves the use of a secret value (Figure 3.2c).
- The secret value itself is not sent; however, if the hash function is not one way, an attacker can easily discover the secret value: If the attacker can observe or intercept a transmission, the attacker obtains the message *M and the hash code C -H(SAB||M).*
- The attacker then inverts the hash function to obtain *SAB||M =H -1(C). Because the* attacker now has both *M and SAB||M, it is a trivial matter to recover SAB.*

- The second preimage resistant property guarantees that it is impossible to find an alternative message with the same hash value as a given message.
- This prevents forgery when an encrypted hash code is used (Figures 3.2a and b).
- If this property were not true, an attacker would be capable of the following sequence:
- First, observe or intercept a message plus its encrypted hash code;
- second, generate an unencrypted hash code from the message;
- third, generate an alternate message with the same hash code.

- A hash function that satisfies the first five properties in the preceding list is referred to as a weak hash function.
- If the sixth property is also satisfied, then it is referred to as a strong hash function.
- The sixth property, collision resistant, protects against a sophisticated class of attack known as the birthday attack.
- In addition to providing authentication, a message digest also provides data integrity.
- It performs the same function as a frame check sequence: If any bits in the message are accidentally altered in transit, the message digest will be in error.

Preimage resistant	2^n
Second preimage resistant	2^n
Collision resistant	$2^{n/2}$

Simple Hash Functions

- All hash functions operate using the following general principles. The input (message, file, etc.) is viewed as a sequence of *n-bit blocks. The input is processed one* block at a time in an iterative fashion to produce an *n-bit hash function.*
- One of the simplest hash functions is the bit-by-bit exclusive-OR (XOR) of every block.
- This can be expressed as

$$C_i = b_{i1} \oplus b_{i2} \oplus \ldots \oplus b_{im}$$

where

C_i = ith bit of the hash code, $1 \le i \le n$
m = number of n-bit blocks in the input
b_{ij} = ith bit in jth block
$\oplus$ = XOR operation

- Figure 3.3 illustrates this operation; it produces a simple parity for each bit position and is known as a longitudinal redundancy check.
- It is reasonably effective for random data as a data integrity check. Each *n-bit hash value is equally likely*.
- Thus, the probability that a data error will result in an unchanged hash value is 2 *n*.
- With more predictably formatted data, the function is less effective.
- A simple way to improve matters is to perform a 1-bit circular shift, or rotation, on the hash value after each block is processed.

- The procedure can be summarized as

1. Initially set the *n-bit hash value to zero.*

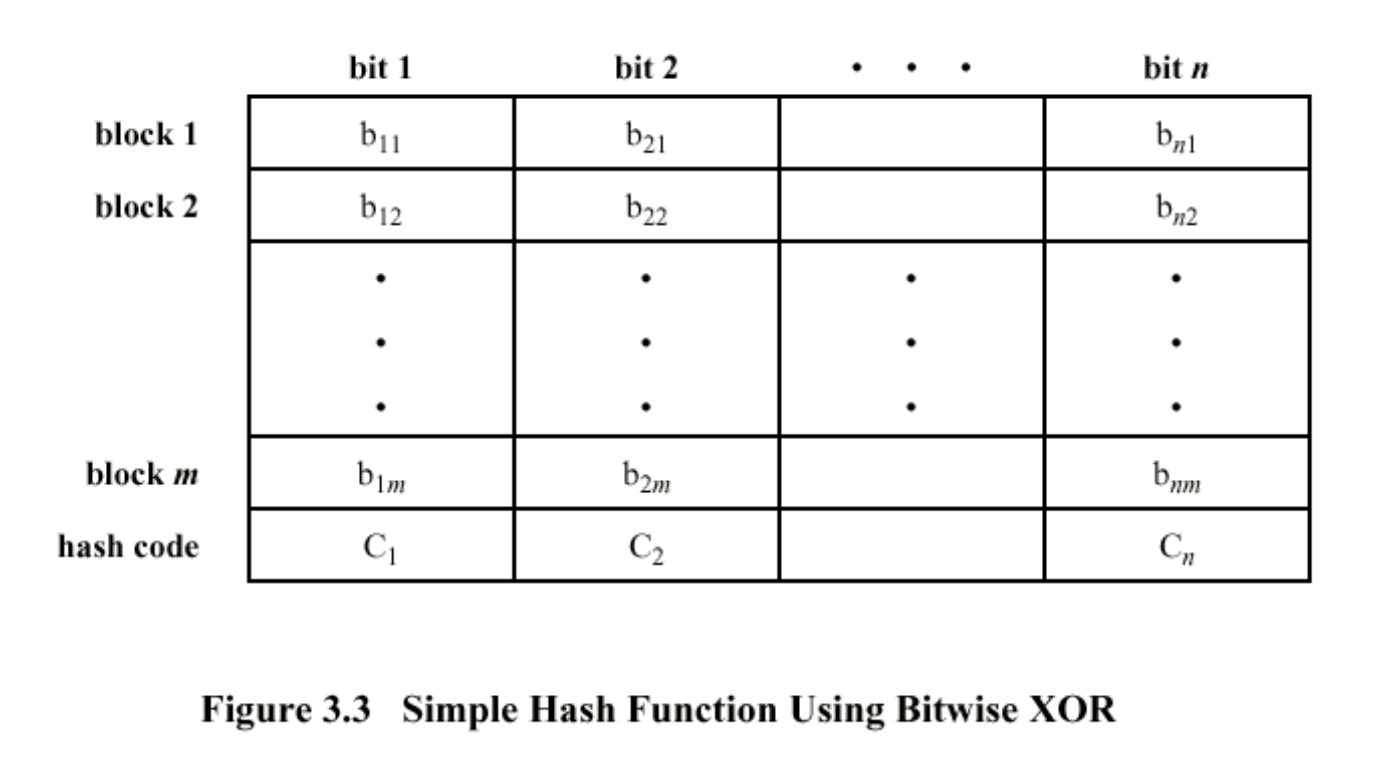
2. Process each successive *n-bit block of data:*

a. Rotate the current hash value to the left by one bit.

b. XOR the block into the hash value.

- This has the effect of "randomizing" the input more completely and overcoming any regularities that appear in the input.

Simple Hash Function

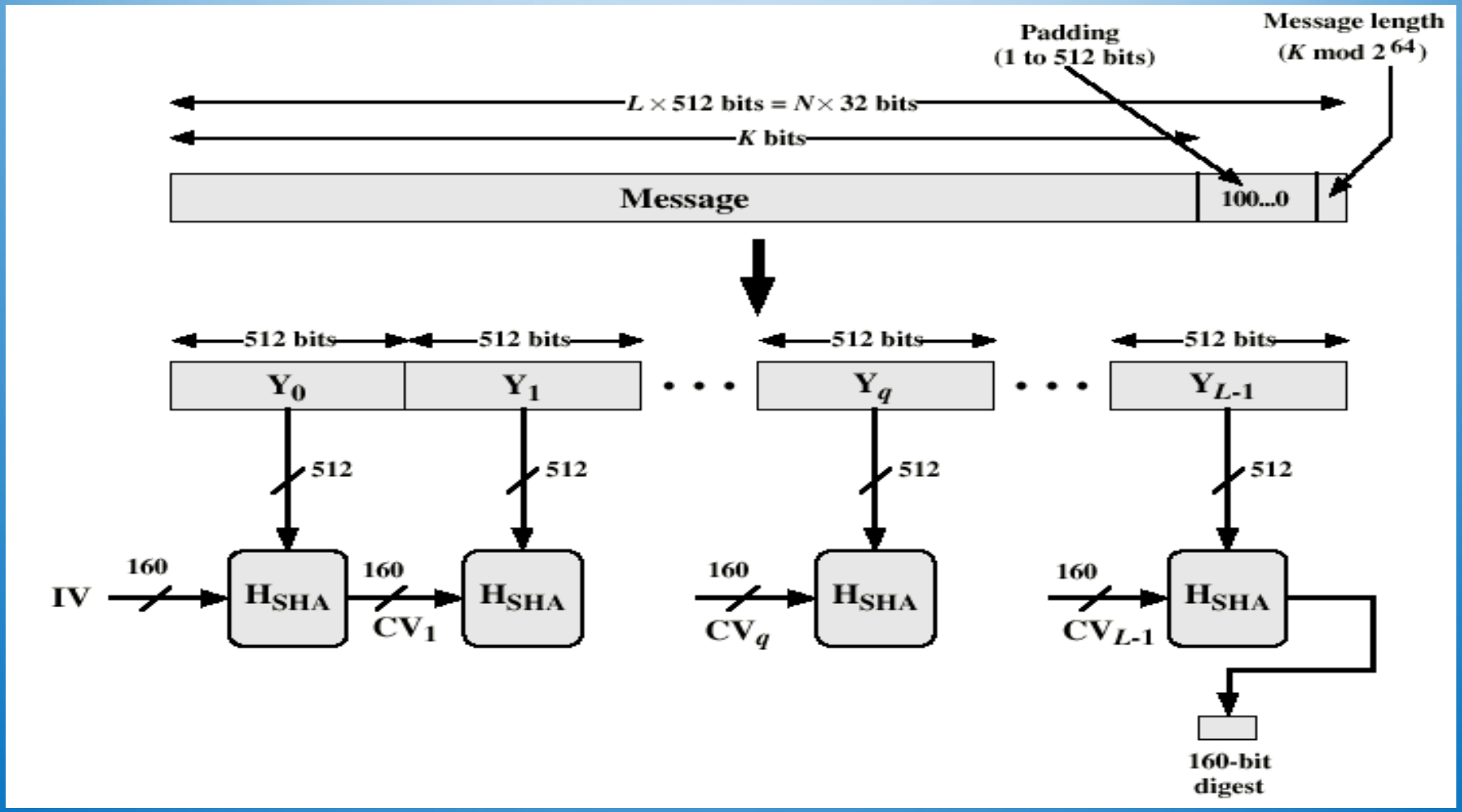


	bit 1	bit 2	• • •	bit *n*
block 1	b_{11}	b_{21}		b_{n1}
block 2	b_{12}	b_{22}		b_{n2}
	•	•	•	•
	•	•	•	•
	•	•	•	•
block *m*	b_{1m}	b_{2m}		b_{nm}
hash code	C_1	C_2		C_n

Figure 3.3 Simple Hash Function Using Bitwise XOR

- One-bit circular shift on the hash value after each block is processed would improve the code

Message Digest Generation Using SHA-1



- The algorithm takes as input a message with a maximum length of less than 2^128 bits and produces as output a 512-bit message digest.
- The input is processed in 1024-bit blocks.
- Figure 3.4 depicts the overall processing of a message to produce a digest.

The processing consists of the following steps.

Step 1 Append padding bits:

The message is padded so that its length is congruent to 896 modulo 1024. Padding is always added, even if the message is already of the desired length. Thus, the number of padding bits is in the range of 1 to 1024.The padding consists of a single 1 bit followed by the necessary number of 0 bits.

- **Step 2 Append length:**
- A block of 128 bits is appended to the message.
- This block is treated as an unsigned 128-bit integer (most significant byte first) and contains the length of the original message (before the padding).
- The outcome of the first two steps yields a message that is an integer
- multiple of 1024 bits in length. In Figure 3.4, the expanded message is represented
- as the sequence of 1024-bit blocks *M1*, *M2*, . . ., *MN*, *so that the total*
- length of the expanded message is $N \times 1024$ *bits.*

- **Step 3 Initialize hash buffer:**

- A 512-bit buffer is used to hold intermediate and final results of the hash function.
- The buffer can be represented as eight 64-bit registers (*a*, *b*, *c*, *d*, *e*, *f*, *g*, *h*).*These registers are initialized to the following 64-bit* integers (hexadecimal values):

SHA-1 Processing of single 512-Bit Block

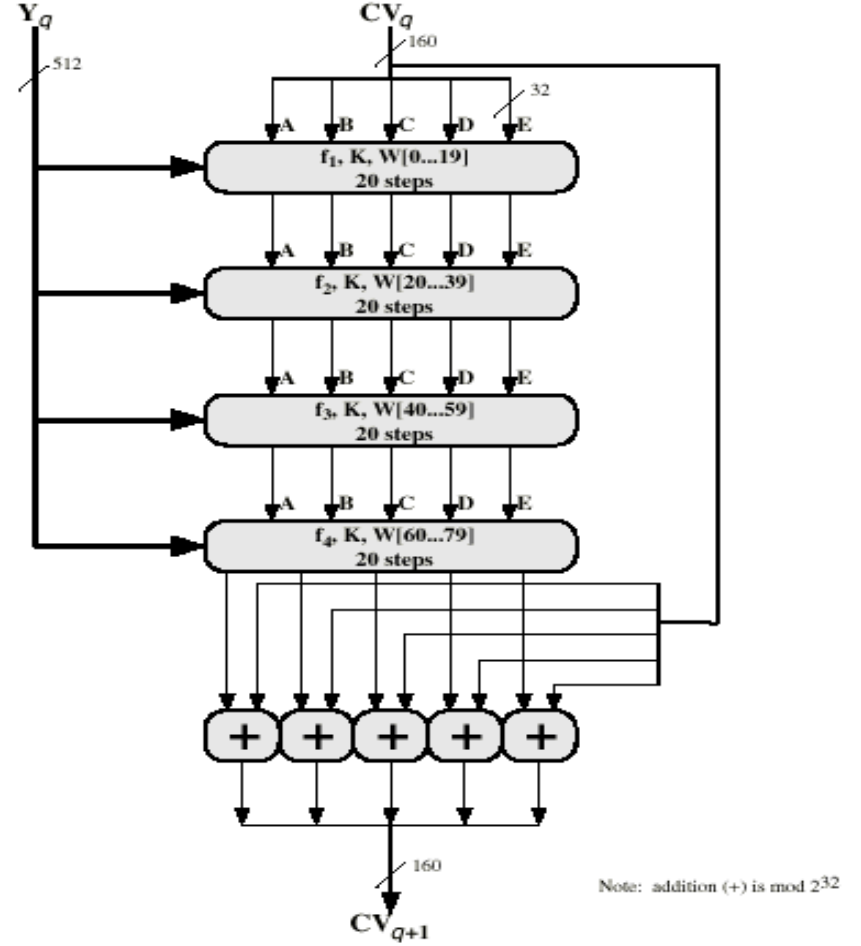


Figure 3.5 SHA-1 Processing of a Single 512-bit Block

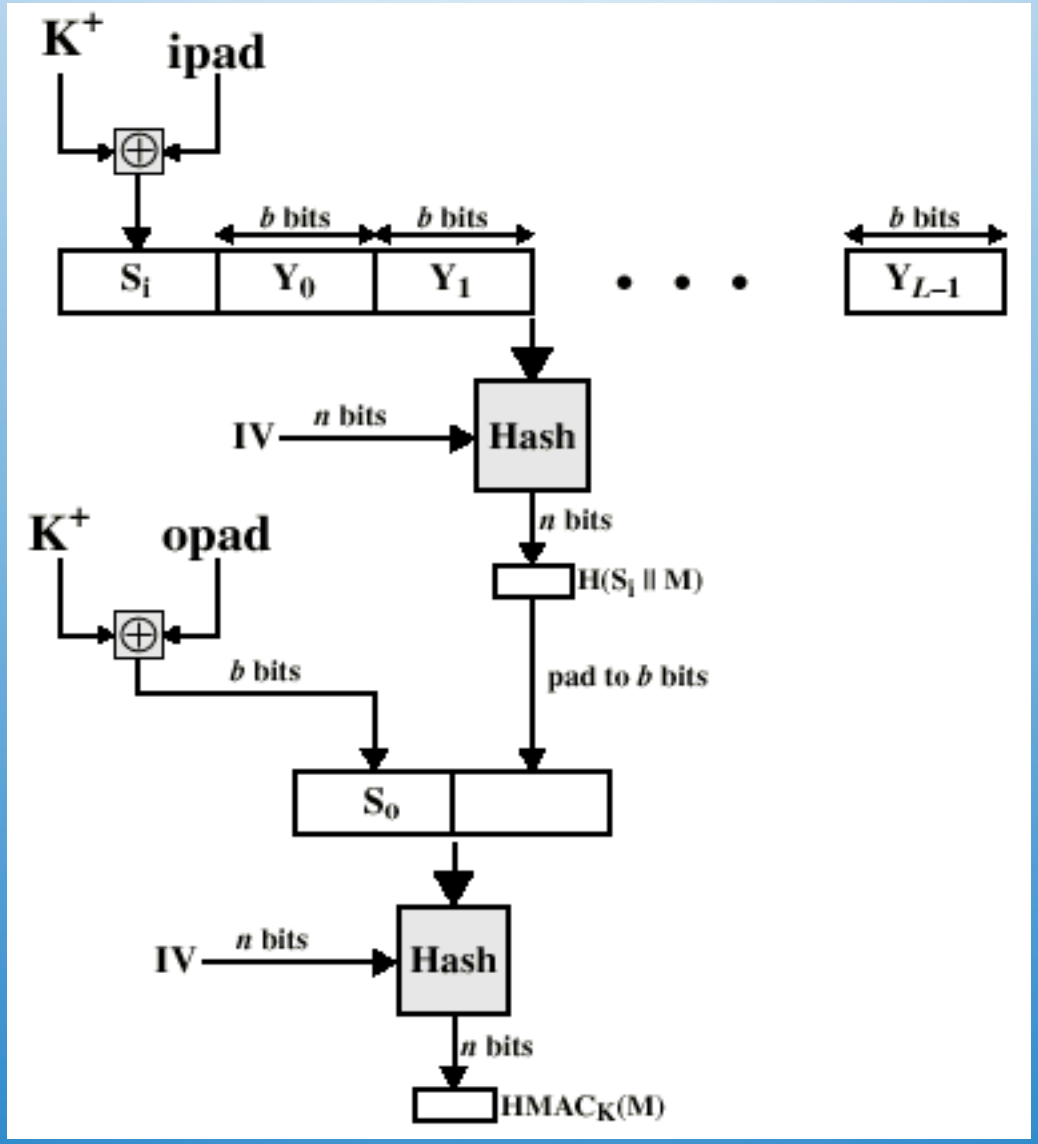
Other Secure HASH functions

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

HMAC

- Use 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 (Not a problem anymore)

HMAC Structure



HMAC DESIGN OBJECTIVES:

- To use, without modifications, available hash functions. In particular, hash functions that perform well in software, and for which code is freely and widely available
- To allow for easy replaceability of the embedded hash function in case faster or more secure hash functions are found or required
- To preserve the original performance of the hash function without incurring a significant degradation
- To use and handle keys in a simple way
- To have a well-understood cryptographic analysis of the strength of the authentication mechanism based on reasonable assumptions on the embedded hash function

HMAC ALGORITHM

- H embedded hash function (e.g., SHA-1)
- *M message input to HMAC (including the padding specified in the embedded* hash function)
- *Yi ith block of M, 0 ≤ i ≤ (L 1)*
- *L = number of blocks in M*
- *b number of bits in a block*
- *n length of hash code produced by embedded hash function*
- *K secret key; if key length is greater than b, the key is input to the hash* function to produce an *n-bit key; recommended length is > n*
- *K +padded with zeros on the left so that the result is b bits in length*
- ipad 00110110 (36 in hexadecimal) repeated *b/8 times*
- opad 01011100 (5C in hexadecimal) repeated *b/8 times*

1. Append zeros to the left end of *K to create a b-bit string K (e.g., if K is of* length 160 bits and *b 512, then K will be appended with 44 zero bytes).*

2. XOR (bitwise exclusive-OR) *K + with ipad to produce the b-bit block Si .*

3. Append *M to Si.*

4. Apply H to the stream generated in step 3.

5. XOR *K with opad to produce the b-bit block So.*

6. Append the hash result from step 4 to *So.*

7. Apply H to the stream generated in step 6 and output the result.

- XOR with ipad results in flipping one-half of the bits of *K*.
- Similarly, the XOR with opad results in flipping one-half of the bits of *K*, *but a different* set of bits.
- In effect, by passing *Si and So through the hash algorithm, we have* pseudorandomly generated two keys from *K*.
- HMAC should execute in approximately the same time as the embedded hash function for long messages.
- HMAC adds three executions of the basic hash function