Cryptography and Security

Lecture 1

Mosarrat Jahan
mosarratjahan@cse.du.ac.bd

course Website:
https://sites.google.com/cse.univdhaka.edu/cse-4137/cryptography-and-network-security

Security and Standards Organizations

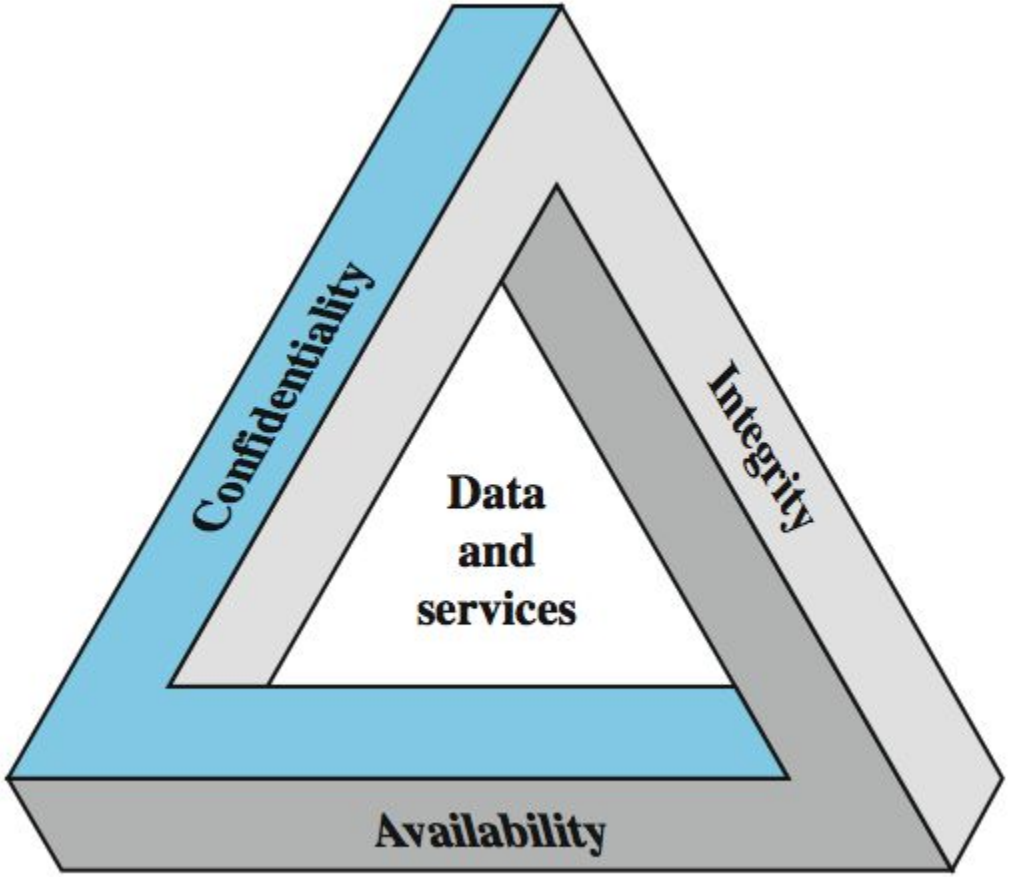
- **National Institute of Standards & Technology (NIST):**
  - NIST FIPS (Federal Information Processing Standard) and SP(Special Publication)
- **Internet Society (ISOC)**
  - Internet Engineering Task Force (IETF)
  - Internet Architecture Board (IAB)
  - Request for Comments (RFC)
- **International Telecommunication Union**
  - Telecommunication Standardization Sector (ITU-T)
  - Recommendation
- **International Organization for Standardization (ISO)**
  - ISO is a nongovernmental organization that promotes the development of standardization and related activities to facilitates the international exchange of goods and services.

Computer Security Concept

According to NIST *Computer Security* Handbook

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).*

Key Security Concepts



Key Security Concepts

- Confidentiality
  - Private information should not be disclosed to unauthorized individuals.
  - Ensures privacy.
- Integrity
  - Guard data against improper modification including information non-repudiation and authenticity.
- Availability
  - Timely and reliable access to and use of information.

Additional Security Concepts

- Authenticity
  - Property to verify the user's originality and that the date arriving is from a trusted user.
- Accountability
  - Ability to trace the action of an entity.
  - This supports non-repudiation, fault isolation, intrusion detection and prevention, after-action recovery and legal actions.

Attacker/Adversary

- “Computer security studies how systems behave in the presence of an adversary.”
- The adversary

  a.k.a. the attacker

  a.k.a. the bad guy
- An intelligence that actively tries to cause the system to misbehave.

Levels of Impacts of Security Breaches

Three levels of impact from a security breach

- Low
- Moderate
- High

Impacts are defined in terms of

- Organizational operation
- Organizational asset
- Financial loss
- Individuals

Examples of Security Requirements

- Confidentiality – Student grades
- Integrity – Patient information
- Availability – Authentication service

Computer Security Challenges

1. not simple
2. must consider potential attacks
3. procedures used counter-intuitive
4. involve algorithms and secret info
5. must decide where to deploy mechanisms
6. battle of wits between attacker / admin
7. not perceived of benefit until fails
8. requires regular monitoring
9. too often an after-thought
10. regarded as impediment to using system

OSI Security Architecture

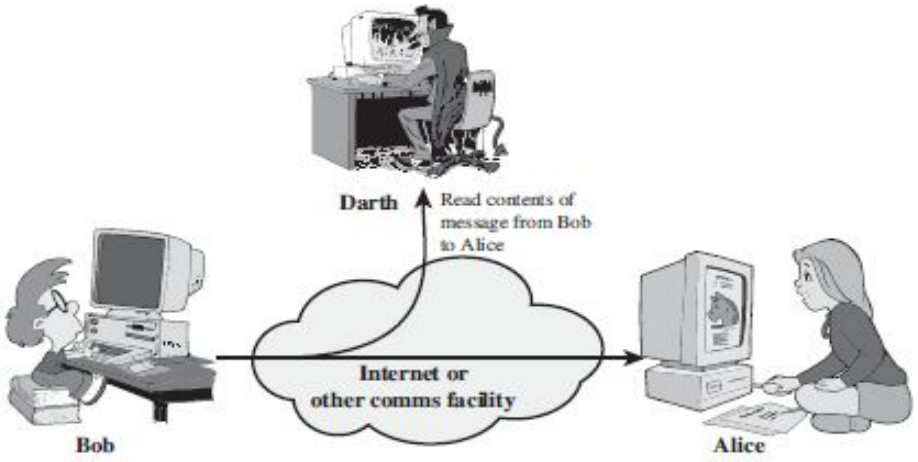
- ITU-T X.800 *"Security Architecture for OSI"*
- Defines a systematic way of defining and providing security requirements
- It provides a useful, if abstract, overview of concepts we will study

OSI Security Architecture

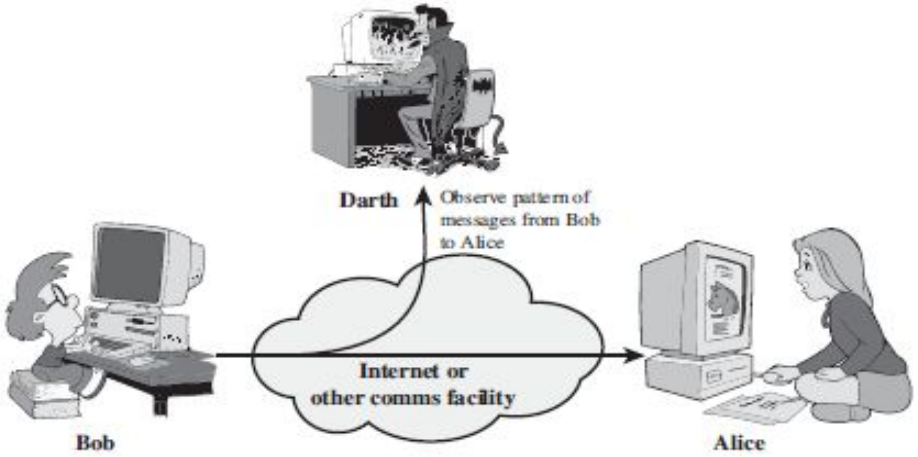
Consider three aspects of information security:

- **security attack**
- **security mechanism**
- **security service**

Passive Attack



(a) Release of message contents

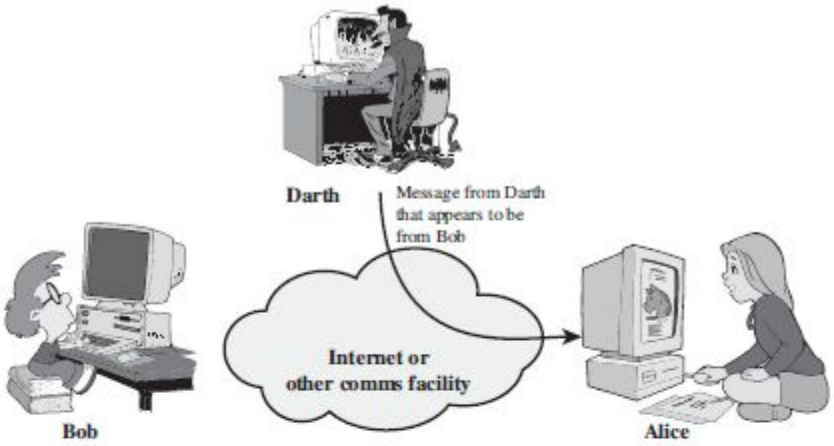


(b) Traffic analysis

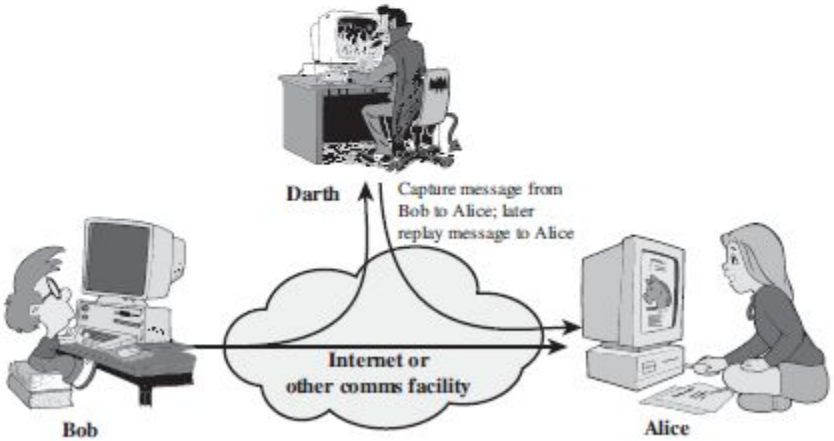
Active Attack

- Modification of data or creation of false data.
- Four types:
  - Masquerade
  - Replay
  - Modification of messages
  - Denial of Services.

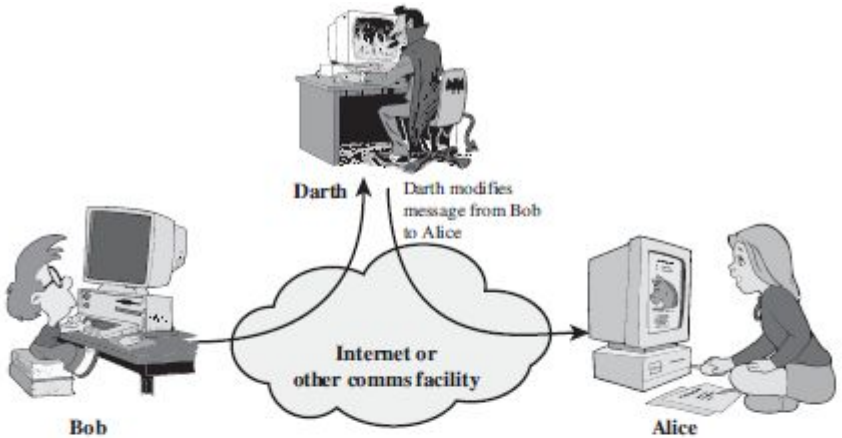
Active Attack



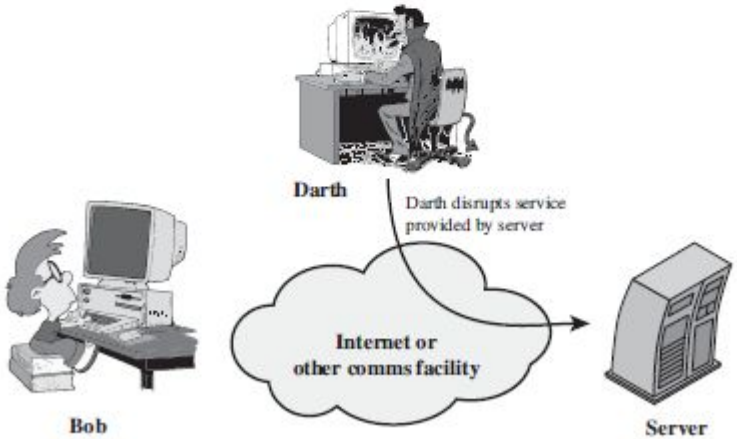
(a) Masquerade



(b) Replay



(c) Modification of messages



(d) Denial of service

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"

- **RFC 2828**:

"a processing or communication service provided by a system to give a specific kind of protection to system resources"

Security Services (X.800)

- **Authentication** - assurance that communicating entity is the one claimed

– have both peer-entity & data origin authentication

- **Access Control** - prevention of the unauthorized use of a resource
- **Data Confidentiality** - protection of data from unauthorized disclosure
- **Data Integrity** - assurance that data received is as sent by an authorized entity
- **Non-Repudiation** - protection against denial by one of the parties in a communication
- **Availability** - resource accessible/usable

Security Services (X.800)

Table 1.2 Security Services (X.800)

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 (i.e., this service controls who can have access to a resource, under what conditions access can occur, and what those accessing the resource are allowed to do).

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 derived from observation of traffic flows.

DATA INTEGRITY

The assurance that data received are exactly as sent by an authorized entity (i.e., contain no modification, insertion, deletion, or replay).

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 within 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, deleted, 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 one of the entities involved in a communication of having participated in all or part of the communication.

Nonrepudiation, Origin
Proof that the message was sent by the specified party.

Nonrepudiation, Destination
Proof that the message was received by the specified party.

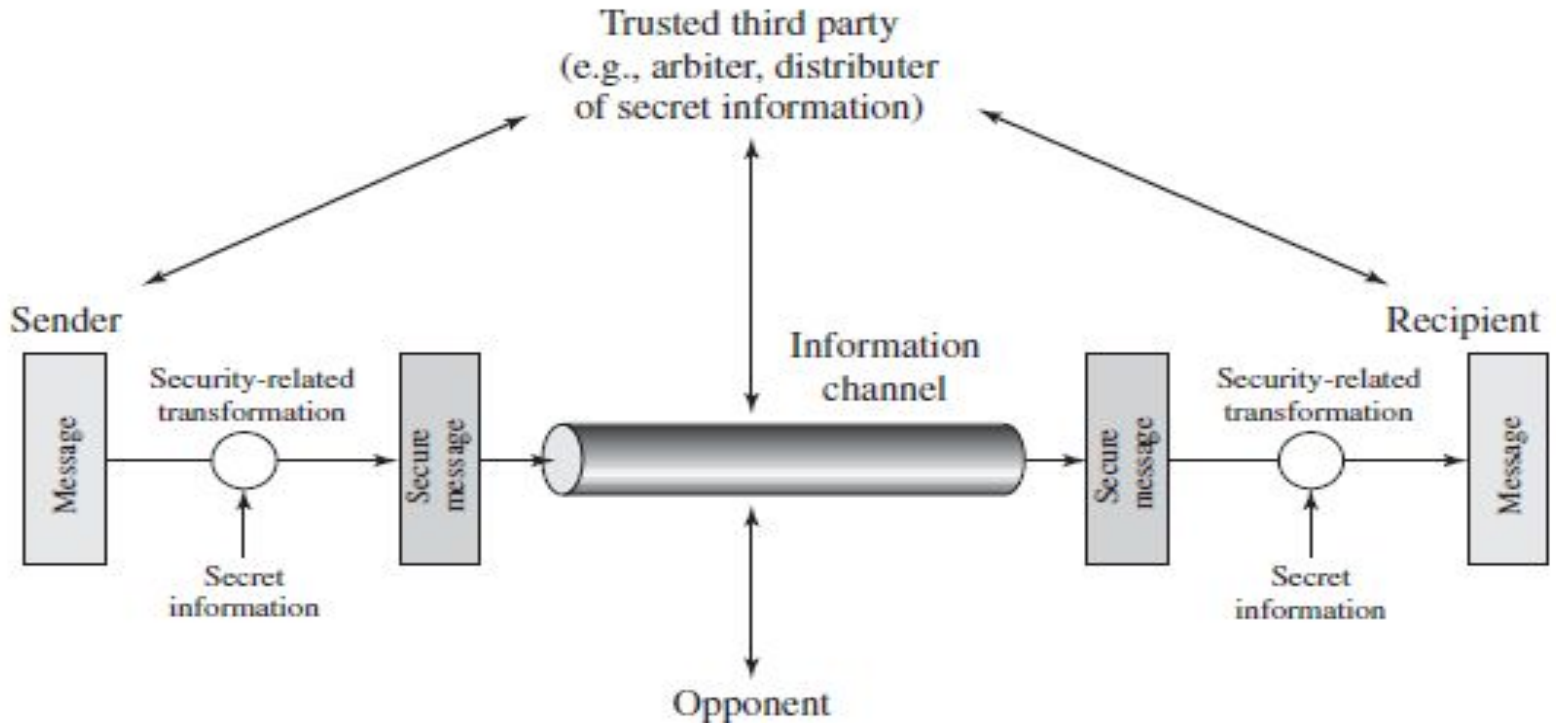
Security Mechanisms

- feature designed to detect, prevent, or recover from a security attack
- no single mechanism that will support all services required
- however one particular element underlies many of the security mechanisms in use:
  - **cryptographic techniques**

Security Mechanisms (X.800)

SPECIFIC SECURITY MECHANISMS	PERVASIVE SECURITY MECHANISMS
May be incorporated into the appropriate protocol layer in order to provide some of the OSI security services.	Mechanisms that are not specific to any particular OSI security service or protocol layer.
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.	**Trusted Functionality** That which is perceived to be correct with respect to some criteria (e.g., as established by a security policy).
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).	**Security Label** The marking bound to a resource (which may be a data unit) that names or designates the security attributes of that resource.
Access Control A variety of mechanisms that enforce access rights to resources.	**Event Detection** Detection of security-relevant events.
Data Integrity A variety of mechanisms used to assure the integrity of a data unit or stream of data units.	**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.
Authentication Exchange A mechanism intended to ensure the identity of an entity by means of information exchange.	**Security Recovery** Deals with requests from mechanisms, such as event handling and management functions, and takes recovery actions.
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.	

Model for Network Security

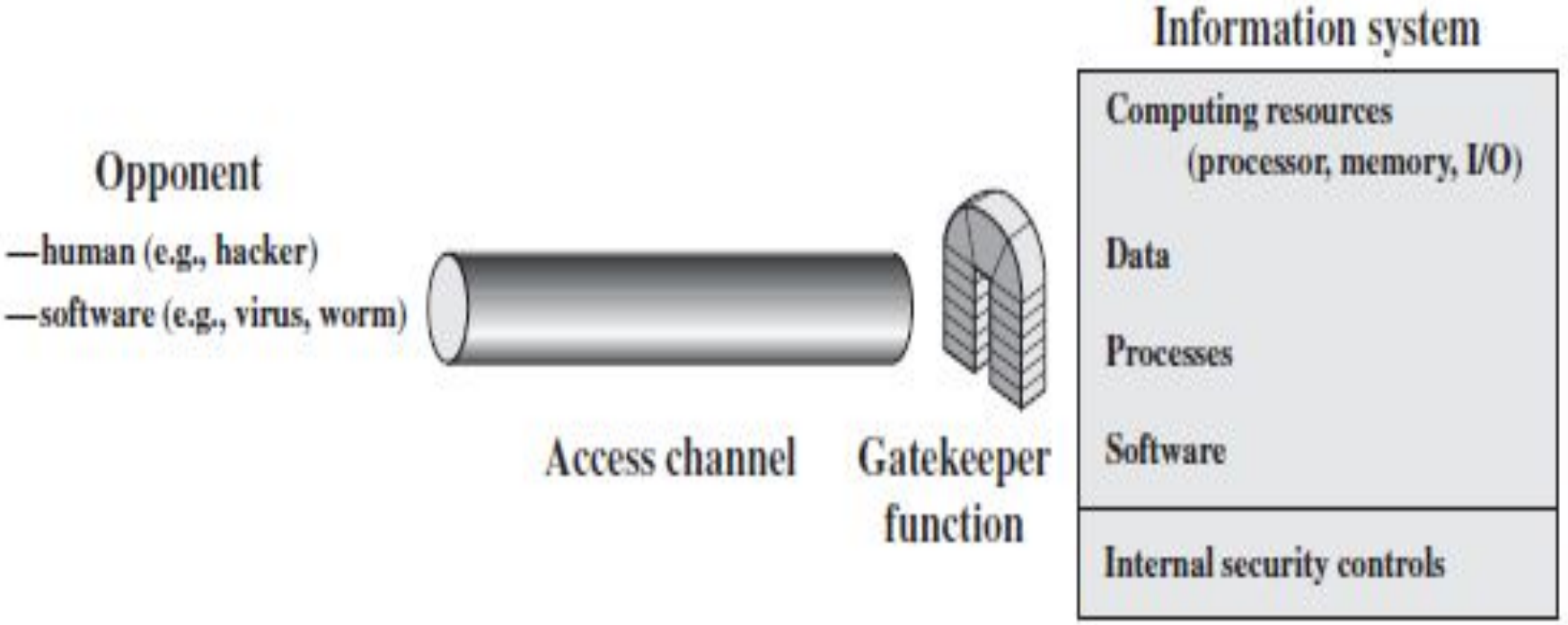


Model for Network Security

This model requires

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

Model for Network Security



Books Followed

- Chapter 1 of Cryptography and Network Security (7^{th} Edition) by William Stallings.
- Chapter 1 of Cryptography and Network Security (2^{nd} Edition) by Behrouz A. Forouzan and Debdeep Mukhopadhyay

Cryptography and Security

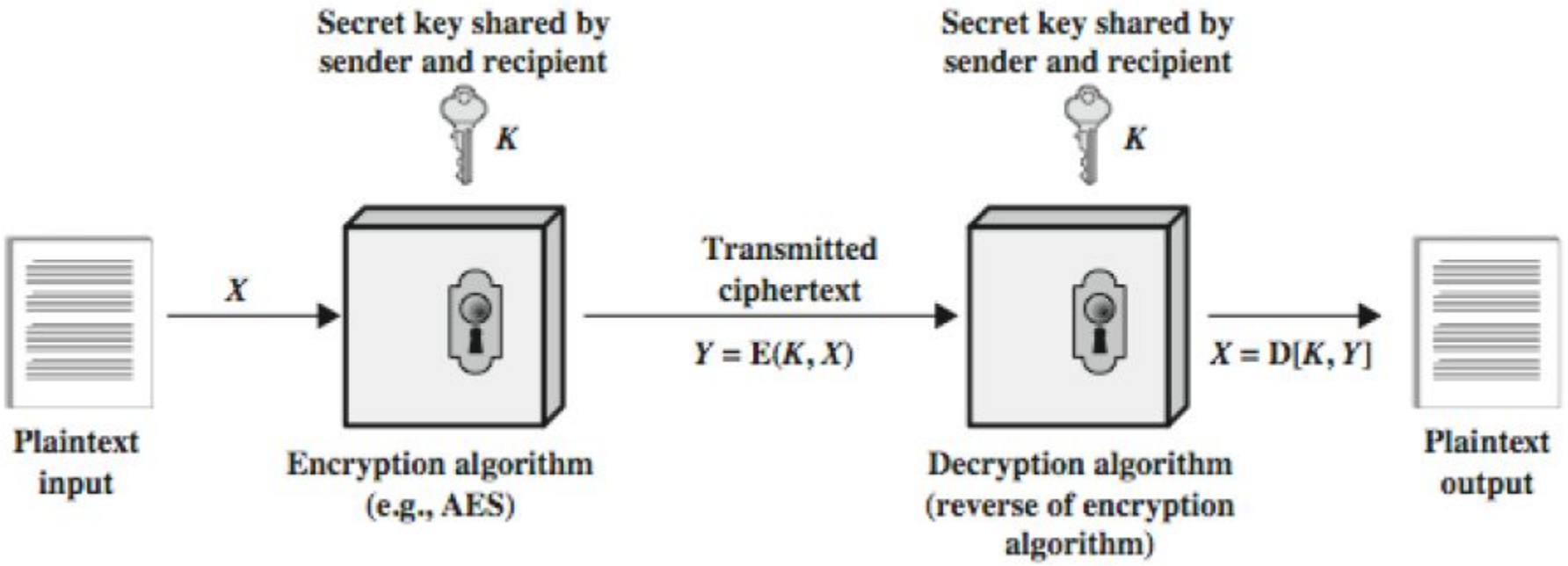
Lecture 2

Classical Encryption Techniques

Some Basic Terminology

- **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 ciphertext from plaintext
- **cryptography**-study of encryption principles/methods
- **cryptanalysis (codebreaking)**-study of principles/ methods of deciphering ciphertext *without* knowing key
- **cryptology**-field of both cryptography and cryptanalysis

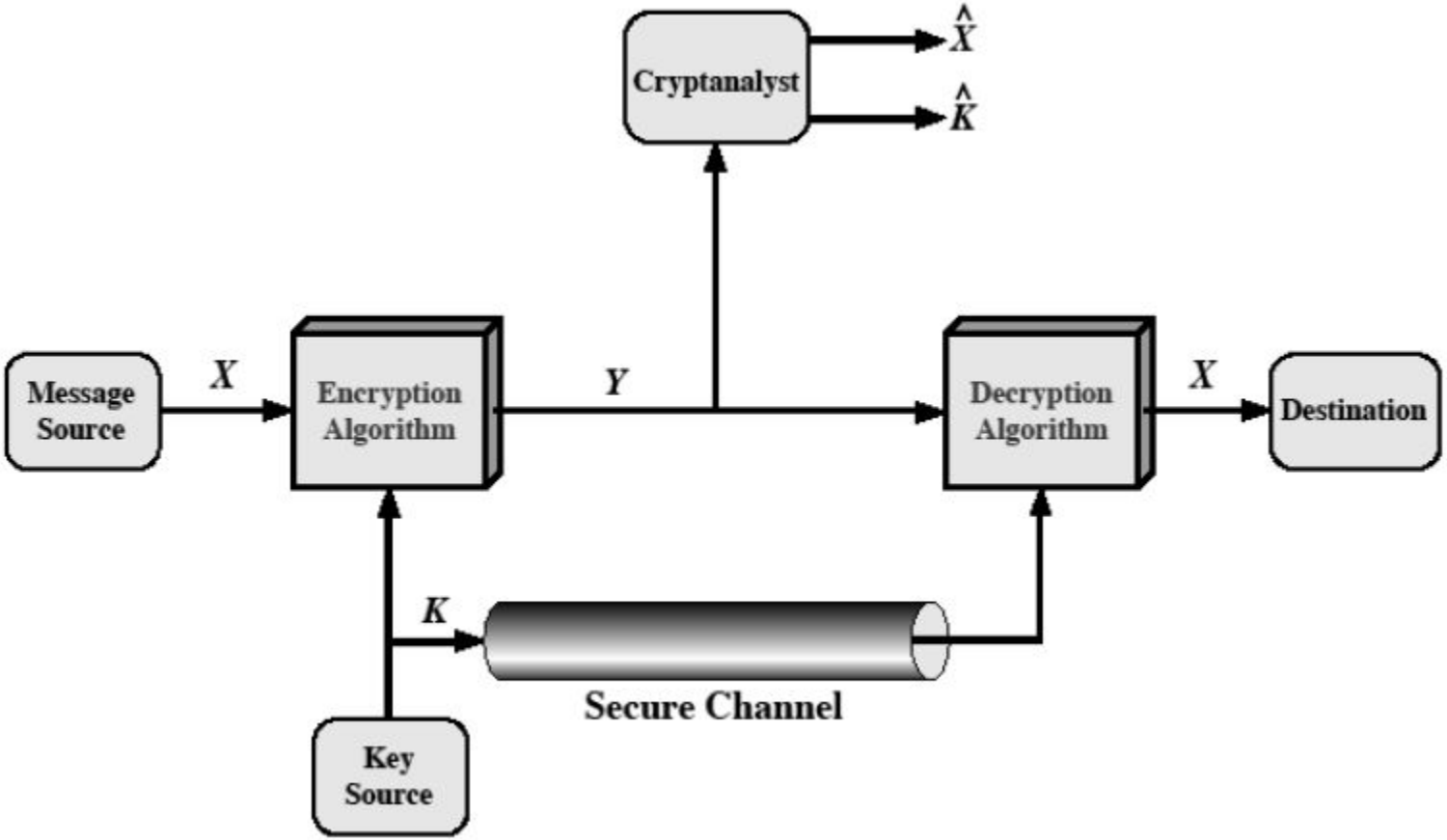
Symmetric Cipher Model



Requirements for Secure Use of Symmetric Key Encryption

1. **Need strong encryption algorithm.**
    - An opponent cannot decrypt a ciphertext if s/he has knowledge of the algorithm, access to some ciphertexts with associated plaintexts without explicit knowledge of key.
    - Algorithms are known.
2. **Maintain the secrecy of the key.**
    - Need secure channel to distribute key.

Model of Symmetric Cryptosystem



Characteristics of Cryptographic System

1. Type of encryption operations used
    - substitution
    - transposition
    - product
2. Number of keys used
    - single-key/ secret-key/symmetric key/ conventional encryption.
    - two-key/ public-key/asymmetric key encryption
3. way in which plaintext is processed
    - block
    - stream

Attacks on Conventional Encryption Scheme

- **Cryptanalysis**
  - Exploits the nature of algorithm based on some knowledge of plaintext or some plaintext-ciphertext pair.
- **Brute-force Attack**
  - Tries every possible keys on a piece of ciphertext.
  - On average, half of all possible keys must be tried.

Types of Cryptanalytic Attacks

Type of Attack	Known to Cryptanalyst
Ciphertext Only	• Encryption algorithm • Ciphertext
Known Plaintext	• Encryption algorithm • Ciphertext • One or more plaintext–ciphertext pairs formed with the secret key
Chosen Plaintext	• Encryption algorithm • Ciphertext • Plaintext message chosen by cryptanalyst, together with its corresponding ciphertext generated with the secret key
Chosen Ciphertext	• Encryption algorithm • Ciphertext • Ciphertext chosen by cryptanalyst, together with its corresponding decrypted plaintext generated with the secret key
Chosen Text	• 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

More Definitions

- **unconditional security**

– no matter how much computer power or time is available, the cipher cannot be broken as the ciphertext provides ***insufficient information*** to uniquely determine the corresponding plaintext.

- **computational security**

- given limited computing resources the cipher cannot be broken.

Brute-Force Attack

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

Table 2.2 Average Time Required for Exhaustive Key Search

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

Substitution Techniques

- where letters of plaintext are replaced by other letters or by numbers or symbols
- or if plaintext is viewed as a sequence of bits, then substitution involves replacing plaintext bit patterns with ciphertext bit patterns

Caesar Cipher

- earliest known substitution cipher
- first attested use in military affairs
- replaces each letter by 3rd letter
- example:

```
plain:  meet me after the toga party
cipher: PHHW PH DIWHU WKH WRJD SDUWB
```

Caesar Cipher

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

- Caesar cipher is defined as:

$$c = E(k, p) = (p + k) \bmod (26)$$

$$p = D(k, c) = (c - k) \bmod (26)$$

Brute-Force Attack on Caesar Cipher

- Brute-force attack on Caesar cipher is possible:
  - The encryption and decryption algorithm are known.
  - For each ciphertext component, only 25 keys to try.
  - The language of plaintext is known and easily recognizable.

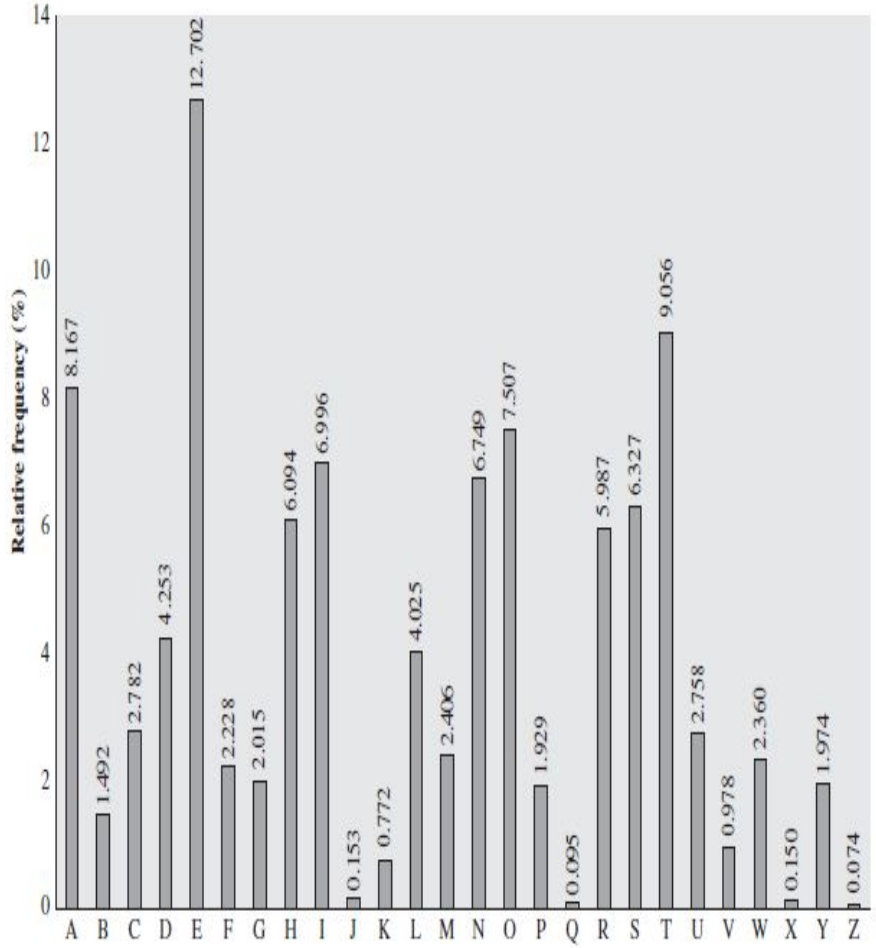
Monoalphabetic Cipher

- A single plain alphabet is mapped to an individual cipher alphabet (per message).
- rather than just shifting the alphabet could shuffle (jumble) the letters arbitrarily

```
plain:  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
cipher: 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
```

- now have a total of 26! = 4 x 10^{26} keys
- with so many keys, might think is secure
- but would be **!!WRONG!!**
- problem is language characteristics

Language Redundancy and Cryptanalysis



- key concept - monoalphabetic substitution ciphers do not change relative letter frequencies.

UZQSOVUOHXMOPVGPOZPEVSGZWSZOPFPESXUDBMETSXAIZ

VUEPHZHMDZSHZOWSFPAPPDTSVPQUZWYMXUZUHSX

EPYEPOPDZSZUFPOMBZWPFUPZHMDJUDTMOHMQ

P 13.33	H 5.83	F 3.33	B 1.67	C 0.00
Z 11.67	D 5.00	W 3.33	G 1.67	K 0.00
S 8.33	E 5.00	Q 2.50	Y 1.67	L 0.00
U 8.33	V 4.17	T 2.50	I 0.83	N 0.00
O 7.50	X 4.17	A 1.67	J 0.83	R 0.00
M 6.67				

Language Redundancy and Cryptanalysis

```
UZQSOVUOHXMOPVGPOZPEVSGZWSZOPFPESXUDBMETSXAIZ
 t a          e   e te   a that e e a          a
VUEPHZHMDZSHZOWSFPAPPDTSVPQUZWYMXUZUHSX
   e t    ta t ha e ee   a e   th     t   a
EPYEPOPDZSZUFPOMBZWPFUPZHMDJUDTMOHMQ
 e   e e tat   e    the    t
```

- Finally the decrypted message is

it was disclosed yesterday that several informal but direct contacts have been made with political representatives of the viet cong in moscow

Playfair Cipher

- not even the large number of keys in a Monoalphabetic cipher provides security.
- one approach to improve security was to encrypt multiple letters
- the Playfair Cipher is an example
- Invented by Charles Wheatstone in 1854.

Playfair Cipher

- a 5X5 matrix of letters based on a keyword
- fill in letters of keyword (no duplicates)
- fill rest of matrix with other letters
- eg. 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

Playfair Cipher

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 with two letters at a time

1. if a pair is a repeated letter, insert filler like 'X'
2. if both letters fall in the same row, replace each with letter to right (wrapping back to start from end)
3. if both letters fall in the same column, replace each with the letter below it (wrapping to top from bottom)
4. otherwise each letter is replaced by the letter in the same row and in the column of the other letter of the pair

Security of Playfair Cipher

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

Security of Ciphers

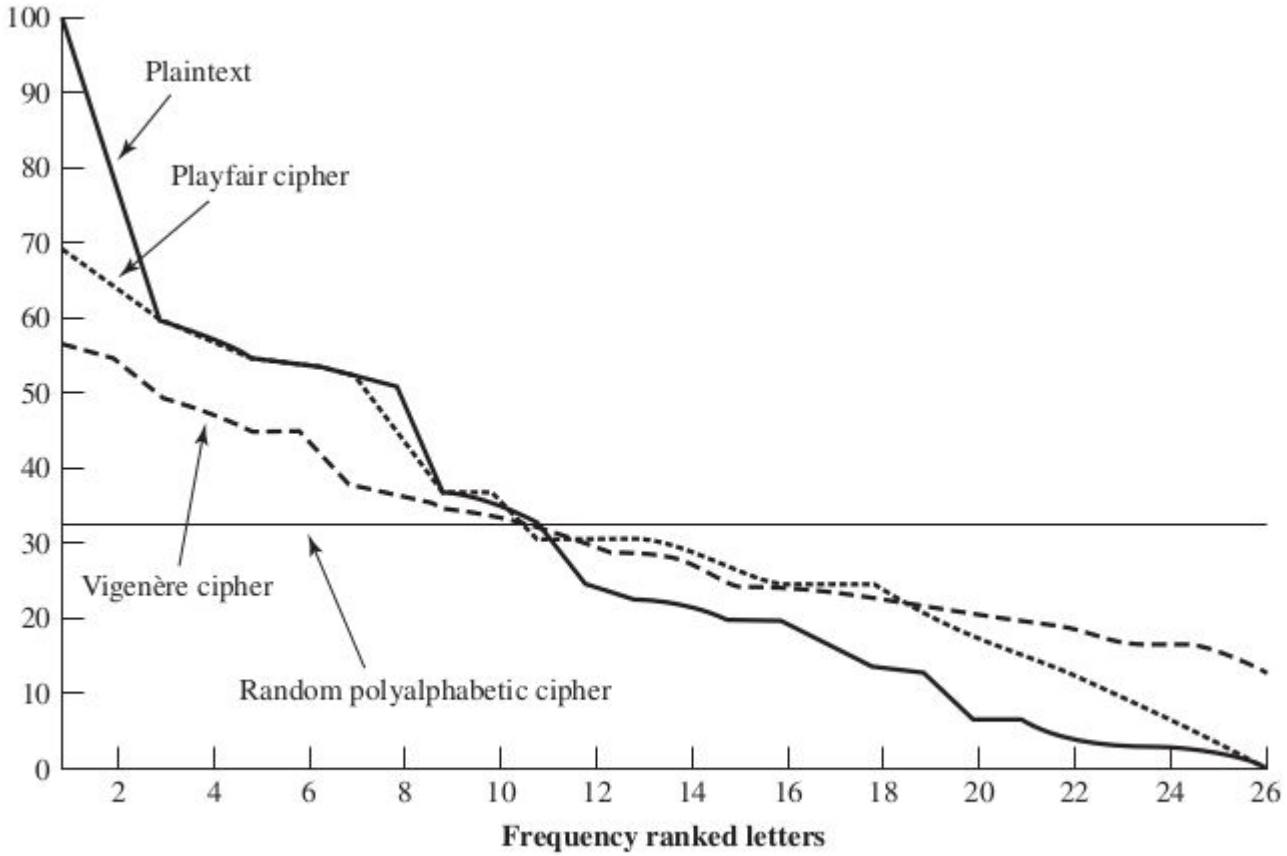


Figure 2.6 Relative Frequency of Occurrence of Letters

Polyalphabetic Ciphers

- improve security using multiple cipher alphabets
- make cryptanalysis harder with more alphabets to guess and flatter frequency distribution
- use a key to select which alphabet is used for each letter of the message
- Common features:
  - A set of related monoalphabetic substitution rules is used.
  - A key determines which particular rule is used for a given transformation.

Vigenère Cipher

- Simplest form of polyalphabetic cipher.
- Plaintext $P=p_0p_1p_2\ldots\ldots p_{n-1}$

  Key $K = k_0k_1k_2\ldots\ldots k_{m-1}$, $m<n$

  Ciphertext $C =C_0C_1C_2\ldots\ldots C_{n-1}$ is expressed as follows:

$$C = C_0, C_1, C_2, \ldots, C_{n-1} = \mathrm{E}(K, P) = \mathrm{E}[(k_0, k_1, k_2, \ldots, k_{m-1}), (p_0, p_1, p_2, \ldots, p_{n-1})]$$
$$= (p_0 + k_0) \bmod 26, (p_1 + k_1) \bmod 26, \ldots, (p_{m-1} + k_{m-1}) \bmod 26,$$
$$(p_m + k_0) \bmod 26, (p_{m+1} + k_1) \bmod 26, \ldots, (p_{2m-1} + k_{m-1}) \bmod 26, \ldots$$

- The general equation for encryption

$$C_i = (p_i + k_{i \bmod m}) \bmod 26$$

- The general equation for decryption

$$p_i = (C_i - k_{i \bmod m}) \bmod 26$$

Vigenère Cipher

- Example:

```
key:        deceptivedeceptivedeceptive
plaintext:  wearediscoveredsaveyourself
ciphertext:ZICVTWQNGRZGVTWAVZHCQYGLMGJ
```

key	3	4	2	4	15	19	8	21	4	3	4	2	4	15
plaintext	22	4	0	17	4	3	8	18	2	14	21	4	17	4
ciphertext	25	8	2	21	19	22	16	13	6	17	25	6	21	19

key	19	8	21	4	3	4	2	4	15	19	8	21	4
plaintext	3	18	0	21	4	24	14	20	17	18	4	11	5
ciphertext	22	0	21	25	7	2	16	24	6	11	12	6	9

Security of Vigenère Ciphers

- have multiple ciphertext letters for each plaintext letter, obscuring letter frequency
- Considerable frequency information exists.

Attacking Vigenère Ciphers

- start with letter frequencies
  - see if they look monoalphabetic or not
- if not, then check for Vigenere cipher.

Kasiski Method:

- method developed by Babbage / Kasiski
- repetitions in ciphertext give clues to period
- of course, could also be random fluke
- Example from previous slide.
- suggests size of 3 or 9
- then attack each monoalphabetic cipher individually

Autokey System

- ideally want a key as long as the message
- Vigenère proposed the **autokey** cipher where keyword is prefixed to message as key
- eg. given key *deceptive*

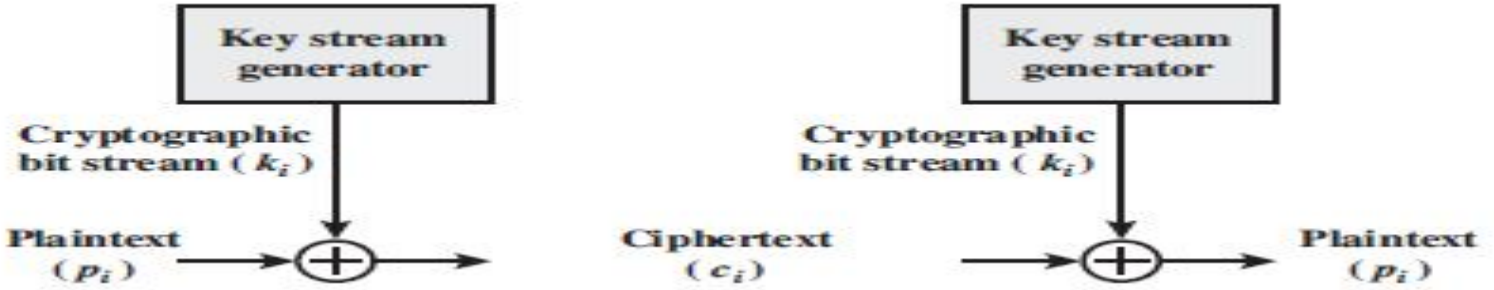
```
key:            deceptivewearediscoveredsav
plaintext:      wearediscoveredsaveyourself
ciphertext:     ZICVTWQNGKZEIIGASXSTSLVVWLA
```

- but can still attack frequency characteristics

Vernam Cipher

- ultimate defense is to use a key as long as the plaintext.
- with no statistical relationship to it
- invented by AT&T engineer Gilbert Vernam in 1918
- originally proposed using a very long but eventually repeating key

Vernam Cipher



- Equation for encryption:

$$c_i = p_i \oplus k_i$$

- Equation for decryption:

$$p_i = c_i \oplus k_i$$

One-Time Pad

- a truly random key as long as the message is used, that makes the cipher unbreakable.
- The key is used only once.
- for **any plaintext** & **any ciphertext** there exists a key mapping one to other
- is unbreakable since ciphertext bears no statistical relationship to the plaintext
- problems in generation & safe distribution of key
- One-time pad has ***perfect secrecy***.

Transposition Ciphers

- known as classical **transposition/permutation** cipher.
- hide the message by rearranging the letter order without altering the actual letters used
- can recognise these since have the same frequency distribution as the original text

Rail Fence cipher

- write message letters out diagonally over a number of rows and then read off cipher row by row.
- Example: Original message

  meet me after the toga party

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

- The ciphertext is

  MEMATRHTGPRYETEFETEOAAT

Row Transposition Ciphers

- is a more complex transposition
- write letters of message out in rows over a specified number of columns
- then reorder the columns according to some key before reading off the rows

```
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:   TTNAAPTMTSUOAODWCOIXKNLYPETZ
```

Product Cipher

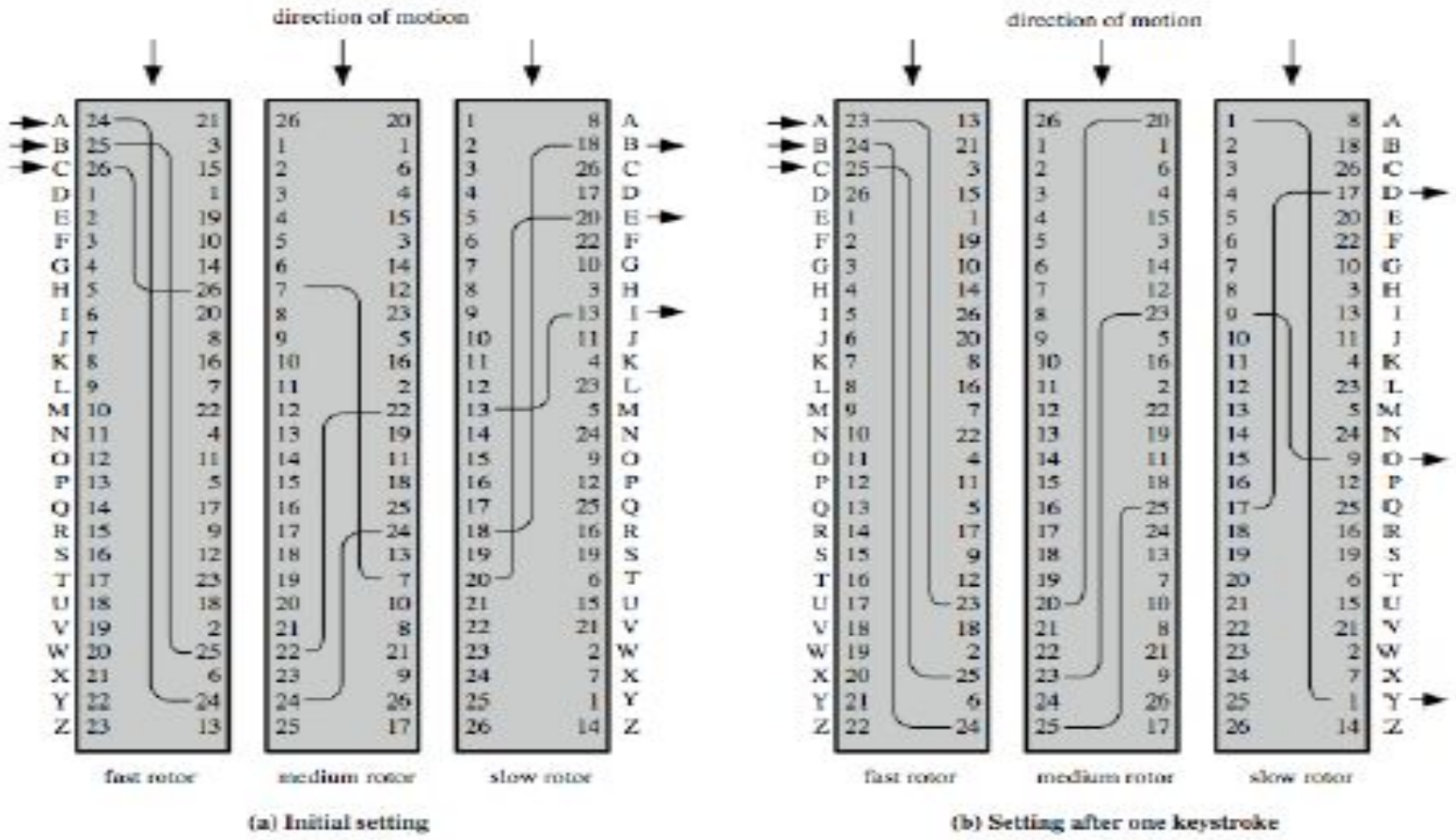
- ciphers using substitutions or transpositions are not secure because of language characteristics
- hence consider using several ciphers in succession to make harder:
  - two substitutions make a more complex substitution
  - two transpositions make more complex transposition
  - but a substitution followed by a transposition makes a new much harder cipher

Rotor Machine

- before modern ciphers, rotor machines were most common complex ciphers in use
- widely used in WW2
  - German Enigma, Allied Hagelin, Japanese Purple
- implemented a very complex, varying substitution cipher
- used a series of cylinders, each giving one substitution, which rotated and changed after each letter was encrypted
- with 3 cylinders have 26^3=17576 alphabets

Hagelin Rotor Machine

Rotor Machine Principles



- **Classical Encryption Techniques from the book of William Stallings**

Cryptography and Security

Lecture 3

Recalling Discrete Probability and One Time Pad

Lecture slides are adopted from slides of Dan Boneh

Discrete Probability

- Finite set $U = \{0,1\}^n$
- **Probability distribution** P over U:

  A function $P: U \longrightarrow [0,1]$ such that $\Sigma\, P(x) = 1$

  where $x \in U$.

  Examples:

  1. Uniform distribution: for all $x \in U$: $P(x) = 1/|U|$
  2. Point distribution at x_0: $P(x_0) = 1$, $\forall\, x \neq x_0$: $P(x) = 0$

- Distribution vector: (P(000), P(001), P(010), … , P(111))

Discrete Probability

- **Event**

  For a set $A \subseteq U$: $\Pr[A] = \Sigma P(x) \in [0,1]$ where $x \in A$ and $\Pr[U]=1$.

  Example: $U = \{0,1\}^8$

- $A = \{$ all x in U such that lsb2(x)=11 $\} \subseteq U$

  for the uniform distribution on $\{0,1\}^8$:

  $\Pr[A] = 1/4$

Discrete Probability

- **Random Variable**

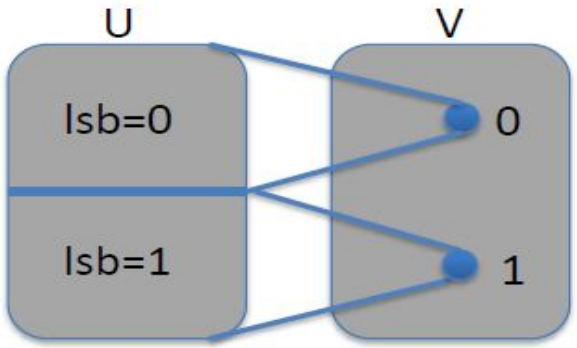
  X is a function $X:U \longrightarrow V$

  Example: $X: \{0,1\}^n \longrightarrow \{0,1\}$;

  $X(y) = lsb(y) \in \{0,1\}$

  For the uniform distribution on U:

  $Pr[X=0] = 1/2 , Pr[X=1] = 1/2$



Discrete Probability

- **Uniform Random Variable**

  Let U be some set, e.g. $U = \{0,1\}^n$

- We write $r \leftarrow U$ to denote a **uniform random variable** over U

  for all $a \in U$: $Pr[r = a] = 1/|U|$

(formally, r is the identity function: $r(x)=x$ for all $x \in U$)

Discrete Probability

Let r be a uniform random variable on $\{0,1\}^2$

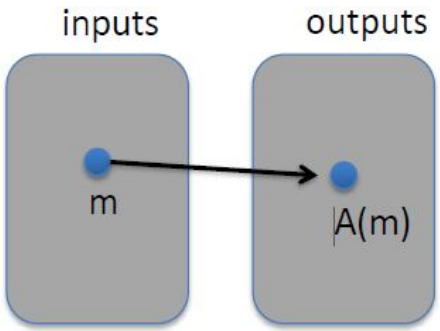
- Define the random variable X = r1 + r2

  Then Pr[X=2] = ¼

  Hint: Pr[X=2] = Pr[r=11]

Discrete Probability

- **Deterministic algorithm**: $y \leftarrow A(m)$

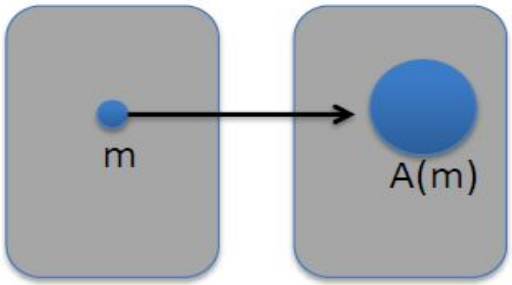


- **Randomized algorithm**

  $y \leftarrow A(m ; r)$ where $r \leftarrow \{0,1\}^{n_R}$

output is a random variable $y \leftarrow A(m)$

- Example: $A(m ; k) = E(k, m)$, $y \xleftarrow{R} A(m)$ ^{R}



Discrete Probability

- **Independence**

  events A and B are **independent** if

  $$\mathbf{Pr[\ A \text{ and } B\ ] = Pr[A] \cdot Pr[B]}$$

- random variables X,Y taking values in V are **independent** if

  $$\mathbf{\forall a,b \in V:\ Pr[\ X=a \text{ and } Y=b] = Pr[X=a] \cdot Pr[Y=b]}$$

- **Example**: $U = \{0,1\}^2 = \{00, 01, 10, 11\}$ and $r \xleftarrow{R} U$

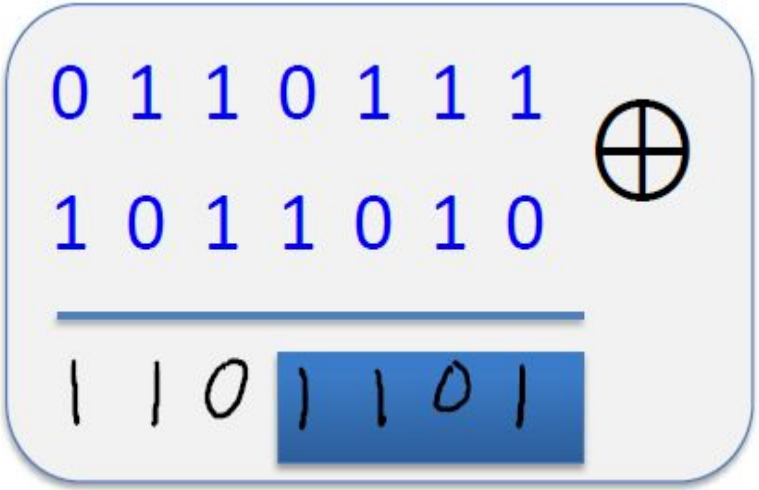
  Define r.v. X and Y as: X = lsb(r) , Y = msb(r)

  $Pr[\ X=0 \text{ and } Y=0\ ] = Pr[\ r=00\ ] = ¼ = Pr[X=0] \cdot Pr[Y=0]$

Review: XOR

- XOR of two strings in $\{0,1\}^n$ is their bit-wise addition mod 2

x	y	$x \oplus y$
0	0	0
0	1	1
1	0	1
1	1	0



An important property of XOR

- **Thm**: Y a rand. var. over $\{0,1\}^n$, X an indep. uniform var. on $\{0,1\}^n$, then Z := Y⊕X is uniform var. on $\{0,1\}^n$

Proof: (for n=1)

$\Pr[Z=0] = \Pr[(x,y)=(0,0) \text{ or } (x,y)=(1,1)] =$

$= \Pr[(x,y)=(0,0)] + \Pr[(x,y)=(1,1)] =$

$= \frac{P_0}{2} + \frac{P_1}{2} = \frac{1}{2}$

■

Y	Pr
0	P_0
1	P_1

X	Pr
0	1/2
1	1/2

X	Y	Pr
0	0	$P_0/2$ ⇐
0	1	$P_1/2$
1	0	$P_0/2$
1	1	$P_1/2$ ⇐

The Birthday Paradox

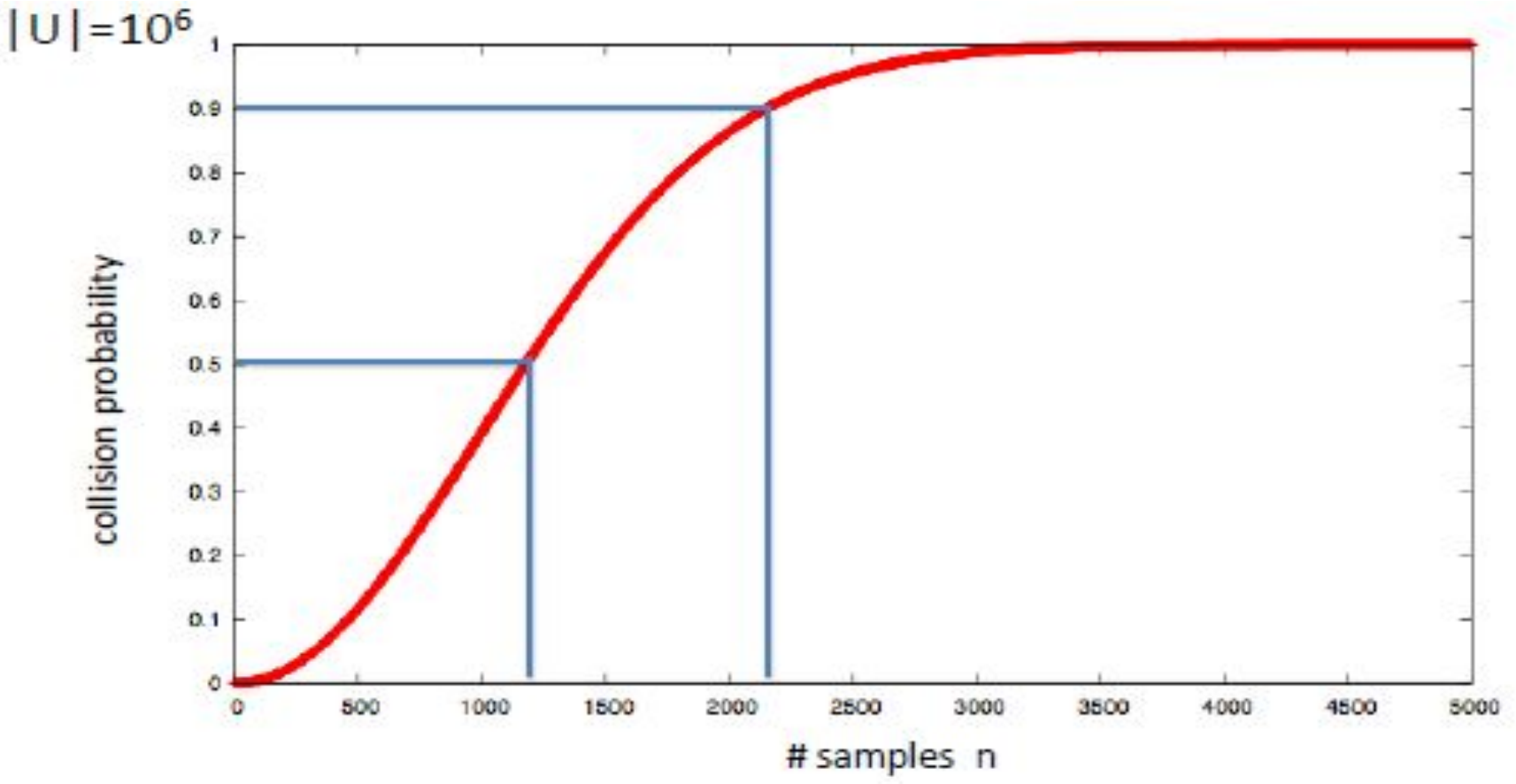
- Let $r_1, \ldots, r_n \in U$ be indep. identically distributed random vars.
- **Thm**:

  when $\mathbf{n} = 1.2 \times |\mathbf{U}|^{1/2}$ then $\Pr[\ \exists\, i \neq j: r_i = r_j\ ] \geq ½$

  Example: Let $U = \{0,1\}^{128}$

  After sampling about 2^{64} random messages from U, some two sampled messages will likely be the same

The Birthday Paradox



Recalling Symmetric Cipher

A **cipher** defined over $(\mathcal{K}, \mathcal{M}, \mathcal{C})$

is a pair of "efficient" algs (***E***, ***D***) where

$E: \mathcal{K} \times \mathcal{M} \rightarrow \mathcal{C}$, $D: \mathcal{K} \times \mathcal{C} \rightarrow \mathcal{M}$

s.t. $\forall m \in \mathcal{M}, k \in \mathcal{K}: \; D(k, E(k, m)) = m$

Where,

E is often randomized.

D is always deterministic.

One Time Pad

First example of a "secure" cipher where

$$\mathcal{M} = \mathcal{C} = \{0,1\}^n \quad , \quad \mathcal{K} = \{0,1\}^n$$

key = (random bit string as long the message)

$$C := E(K, m) = K \oplus m$$

$$D(K, c) = K \oplus c$$

$$D(K, E(K,m)) = D(K, K \oplus m) = K \oplus (K \oplus m) = (K \oplus K) \oplus m = 0 \oplus m = m$$

given a message (*m*) and its OTP encryption (*c*), it is possible to compute the OTP key from *m* and *c* ?

One Time Pad

- Good point: very fast encryption and decryption.
- Bad news: long key (as long as plaintext)

Information Theoretic Security (Shannon 1949)

- CT should reveal no "info" about PT
- A cipher ***(E,D)*** over (K,M,C) has **perfect secrecy** if
  $\forall m_0, m_1 \in M$ ($|m_0| = |m_1|$) and $\forall c \in C$

Pr[E(k,m$_0$)=c] = Pr[E(k,m$_1$)=c] where $k \xleftarrow{R} K$

$\Rightarrow$ Given CT can't tell if msg is m_0 or m_1 (for all m_0, m_1)

$\Rightarrow$ most powerful adv. learns nothing about PT from CT

$\Rightarrow$ no CT only attack!! (but other attacks possible)

One Time Pad (OTP)

- Lemma: OTP has perfect secrecy

Proof:

$$\forall m, c: \quad \Pr_{k}\left[E(k,m)=c\right] = \frac{\#\text{keys } k \in \mathcal{K} \text{ s.t. } E(k,m)=c}{|\mathcal{K}|}$$

e: if $\forall m, c: \quad \#\{k \in \mathcal{K} : E(k,m)=c\} = \text{const.}$

$\Rightarrow$ cipher has perfect secrecy

One Time Pad (OTP)

Let $m \in \mathcal{M}$ and $c \in \mathcal{C}$.

How many OTP keys map m to c ?

None

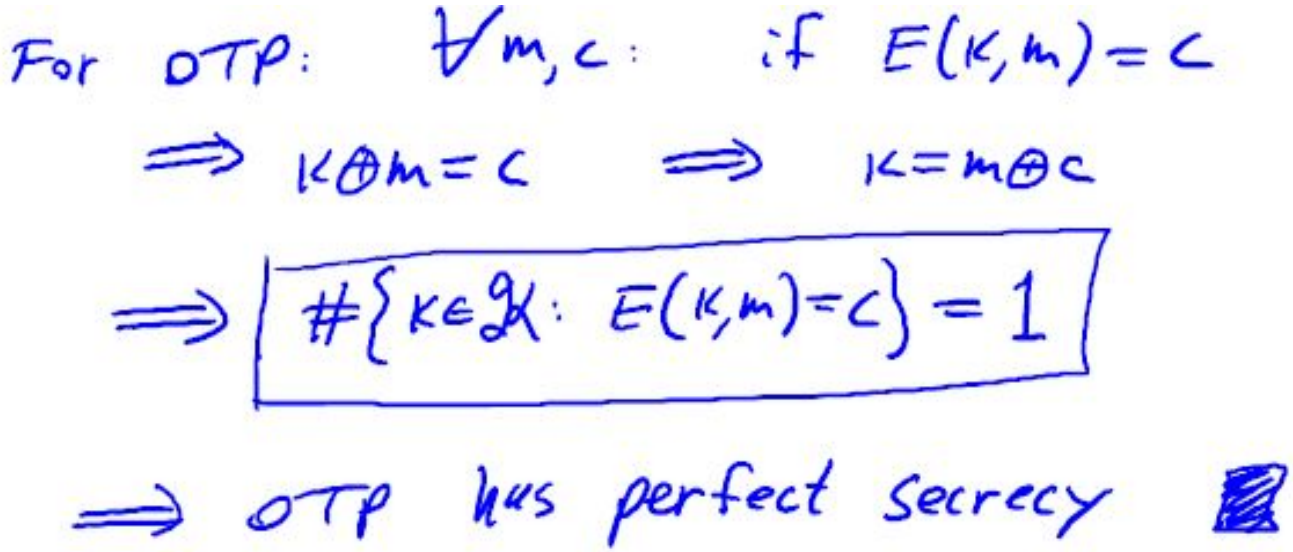
1 

2

Depends on m

One Time Pad (OTP)

- Lemma: OTP has perfect secrecy



OTP: no CT only attack (but other attacks are possible)

One Time Pad (OTP)

- Thm: Perfect secrecy $\Rightarrow$ $|\mathcal{K}| \geq |\mathcal{M}|$
- Implies that ***key length*** >= ***message length***
- Hard to use in practice.

https://crypto.stanford.edu/~dabo/courses/OnlineCrypto/

Cryptography and Security

Lecture 4

Basic Concepts in Number Theory and Finite Fields

Greatest Common Divisor

- A positive integer ***c*** is the greatest common divisor of ***a*** and ***b*** if
  - ***c*** is a divisor of ***a*** and ***b***.
  - Any divisor of ***a*** and ***b*** is a divisor of ***c***.
  - gcd(***a, b***) = max[***k***, such that ***k|a*** and ***k|b***]
- gcd (***a, b***) = gcd(|***a***|,|***b***|)
- gcd(***a,0***)=|***a***|.
- Two integers are relatively prime if their only common positive integer factor is 1. i.e, ***a*** and ***b*** are relatively prime if ***gcd(a,b)=1***.

The Euclidean Algorithm

$$
\begin{aligned}
a &= q_1 b + r_1 && 0 < r_1 < b \\
b &= q_2 r_1 + r_2 && 0 < r_2 < r_1 \\
r_1 &= q_3 r_2 + r_3 && 0 < r_3 < r_2 \\
&\vdots && \vdots \\
r_{n-2} &= q_n r_{n-1} + r_n && 0 < r_n < r_{n-1} \\
r_{n-1} &= q_{n+1} r_n + 0 \\
d &= \gcd(a, b) = r_n
\end{aligned}
$$

- gcd(10,63):

63=10.6+3

10=3.3+1

3=3.1+0

- gcd(1701,3768):

3768=1701.2+366

1701=366.4+237

366=237.1+129

237=129.1+108

129=108.1+21

108=21.5+3→gcd(1701,3768)

21=3.7+0

Modular Arithmetic

- For an integer ***a*** and ***n*** is a positive integer, ***a mod n*** is the remainder when ***a*** is divided by ***n***. The integer ***n*** is called the modulus.

  $a = qn + r \Rightarrow a = floor(a/n) \times n + a \bmod n$

- Two integer ***a*** and ***b*** are said to be **congruent modulo *n***, if $(a \bmod n) = (b \bmod n) \rightarrow a \equiv b \pmod n$

Modular Arithmetic Operations

- Arithmetic operation on the set of integers [0, 1, 2, 3, …, (n-1)].

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. $[(a \bmod n) \times (b \bmod n)] \bmod n = (a \times b) \bmod n$

$11 \bmod 8 = 3; 15 \bmod 8 = 7$
$[(11 \bmod 8) + (15 \bmod 8)] \bmod 8 = 10 \bmod 8 = 2$
$(11 + 15) \bmod 8 = 26 \bmod 8 = 2$

$[(11 \bmod 8) - (15 \bmod 8)] \bmod 8 = -4 \bmod 8 = 4$
$(11 - 15) \bmod 8 = -4 \bmod 8 = 4$

$[(11 \bmod 8) \times (15 \bmod 8)] \bmod 8 = 21 \bmod 8 = 5$
$(11 \times 15) \bmod 8 = 165 \bmod 8 = 5$

Modular Arithmetic Operations

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

$$11^2 = 121 \equiv 4 \pmod{13}$$
$$11^4 = (11^2)^2 \equiv 4^2 \equiv 3 \pmod{13}$$
$$11^7 \equiv 11 \times 4 \times 3 \equiv 132 \equiv 2 \pmod{13}$$

Modulo 8 Addition

+	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

Modulo 8 Multiplication

+	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

Properties of Modular Arithmetic

- ***Z*** = Set of all integers = ***{…, -2, -1, 0, 1, 2, …}***
- Z_n = Set of all non-negative integers less than ***n*** = {***0, 1, 2, …, (n-1)***}
- $Z_2 = \{0, 1\}$
- $Z_8 = \{0, 1, 2, 3, 4, 5, 6, 7\}$
- Z_n = set of residues or residue classes (mod n)
- ***Residue class [r]=[a:a is an integer, a≡r (mod n)***

The residue classes (mod 4) are

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

Properties of Modular Arithmetic in Z_n

Property	Expression
Commutative Laws	$(w + x) \bmod n = (x + w) \bmod n$ $(w \times x) \bmod n = (x + w) \bmod n$
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 $w + z = 0 \bmod n$

Properties of Modular Arithmetic

- **Additive Inverse**

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

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

$$((-a) + a + b) \equiv ((-a) + a + c) \pmod{n}$$
$$b \equiv c \pmod{n}$$

- **Multiplicative Inverse**

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

$$6 \times 3 = 18 \equiv 2 \pmod{8}$$
$$6 \times 7 = 42 \equiv 2 \pmod{8}$$

Yet $3 \neq 7 \pmod{8}$.

If ***a*** and ***n*** have common factor, for a modulus ***n*** and a multiplier ***a*** fails to produce a complete set of residues.

Properties of Modular Arithmetic

With $a = 6$ and $n = 8$,

Z_8	0	1	2	3	4	5	6	7
Multiply by 6	0	6	12	18	24	30	36	42
Residues	0	6	4	2	0	6	4	2

Because we do not have a complete set of residues when multiplying by 6, more than one integer in Z_8 maps into the same residue. Specifically, $6 \times 0 \bmod 8 = 6 \times 4 \bmod 8$; $6 \times 1 \bmod 8 = 6 \times 5 \bmod 8$; and so on. Because this is a many-to-one mapping, there is not a unique inverse to the multiply operation.

However, if we take $a = 5$ and $n = 8$, whose only common factor is 1,

Z_8	0	1	2	3	4	5	6	7
Multiply by 5	0	5	10	15	20	25	30	35
Residues	0	5	2	7	4	1	6	3

The line of residues contains all the integers in Z_8, in a different order.

Euclidean Algorithm Revisited

- For any integer ***a>=0*** and ***b>=0***, gcd(***a, b***) = gcd(***b, a mod b***)
- gcd (55, 22) = gcd(22, 55 mod 22) = gcd (22,11)=11.
- gcd(18,12)=gcd(12, 6)=gcd(6,0)=6.

  - gcd(1701,3768):

3768=1701.2+366 gcd(1701,366)

1701=366.4+237 gcd(366,237)

366=237.1+129 gcd(237,129)

237=129.1+108 gcd(129,108)

129=108.1+21 gcd(108,21)

108=21.5+3 gcd(21,3)

21=3.7+0 gcd(3,0) →gcd(1701,3768)

The Extended Euclidean Algorithm

- Get not only GCD but ***x*** and ***y*** such that ***ax + by*** = ***d*** = GCD(***a,b***)
- useful for latter crypto computations
- follow sequence of divisions for GCD but at each step i, keep track of ***x*** and ***y*** such that ***r = ax + by***
- at the end find GCD value and also ***x*** and ***y***

The Extended Euclidean Algorithm--Example

- gcd(888,54)=6 $\rightarrow$ 6 = 54. (33)+ 888 (-2)

  888 = 54.16+24 $\rightarrow$ 6 = 54 + (888 + 54(-16)) (-2)

  54= 24. 2+ 6 $\rightarrow$ 6= 54+24 (-2)

  24=6. 4 + 0
- gcd(888,54) = 6 = 888***x***+ 54***y*** where ***x*** = -2 and ***y***=33
- Try gcd(56,15) in class; result 15.15+56.(-4)=1

Multiplicative Inverse using Extended Euclidean Algorithm

- Used to find a multiplicative inverse in $\boldsymbol{Z_n}$ for any $\boldsymbol{n}$.
- If extended euclidean algorithm is applied to $\boldsymbol{nx+by} = \boldsymbol{d}$ and the algorithm yields $\boldsymbol{d=1}$, then $\boldsymbol{y = b^{-1}}$ in $\boldsymbol{Z_n}$.
- ***Example:***

  Gcd(56,15) =1 $\rightarrow$ 15.15+56.(-4)=1

  i.e, $\boldsymbol{a}$=56, $\boldsymbol{x}$ =-4, $\boldsymbol{b}$=15 and $\boldsymbol{y}$=15.

  $\boldsymbol{b^{-1}}$=$\boldsymbol{y}$=15 in $\boldsymbol{Z_{56}}$ $\rightarrow$ 15×15 mod 56 = 1.

Group

- **Group**:

A set of elements that is closed with respect to a binary operation denoted by {G, •}.

- Closed ⇒ The result of the operation is also in the set.
- The operation obeys:
  - Closure: if ***a*** and ***b*** belong to G, then ***a.b*** is also in G.
  - Associative law:***(a.b).c = a.(b.c)***
  - Identity element: Has identity ***e***: ***e.a = a.e = a***
  - Inverse element: Has inverses a^{-1}: $a.a^{-1} = e$
- **Abelian Group:**
  - commutative a.b = b.a
  - Example: Z_8, + modular addition, identity =0
  - Order of a finite group is the number of elements in that group.

Cyclic Group

- **Exponentiation:**

  Repeated application of operator

  example: $a^3 = a.a.a$

- **Cyclic Group:**
  - Every element is a power of some fixed element,i.e, $b = a^k$ for some a and every b in group.
  - a is said to be a generator of the group
  - Example: {1, 2, 4, 8} with mod 12 multiplication, the generator is 2.
  - $2^0=1$, $2^1=2$, $2^2=4$, $2^3=8$, $2^4=4$, $2^5=8$
  - A cyclic group is always abelian and may be finite or infinite.
  - The additive group of integers is an infinite cyclic group generated by 1.

Ring

- **Ring**:
  - A set with two operations: addition and multiplication, denoted by {R, +, ×}
  - Abelian group with respect to addition: ***a+b = b+a***
  - Closed under multiplication. ⟶ for ***a, b ϵ R***, ***a.b*** is also in ***R***.
  - Associativity of multiplication. ⟶ ***a.(b.c)=(a.b).c***
  - Multiplication distributes over addition⟶ ***a.(b+c)=a.b+a.c and (a+b).c = a.c + b.c***
- **Commutative Ring:**

  Multiplication is commutative ⟶ ***a.b = b.a***
- **Example:**

  Z_8, +, × is a commutative ring.

Ring

- **Integral Domain:**

  multiplication operation has an identity and no zero divisors

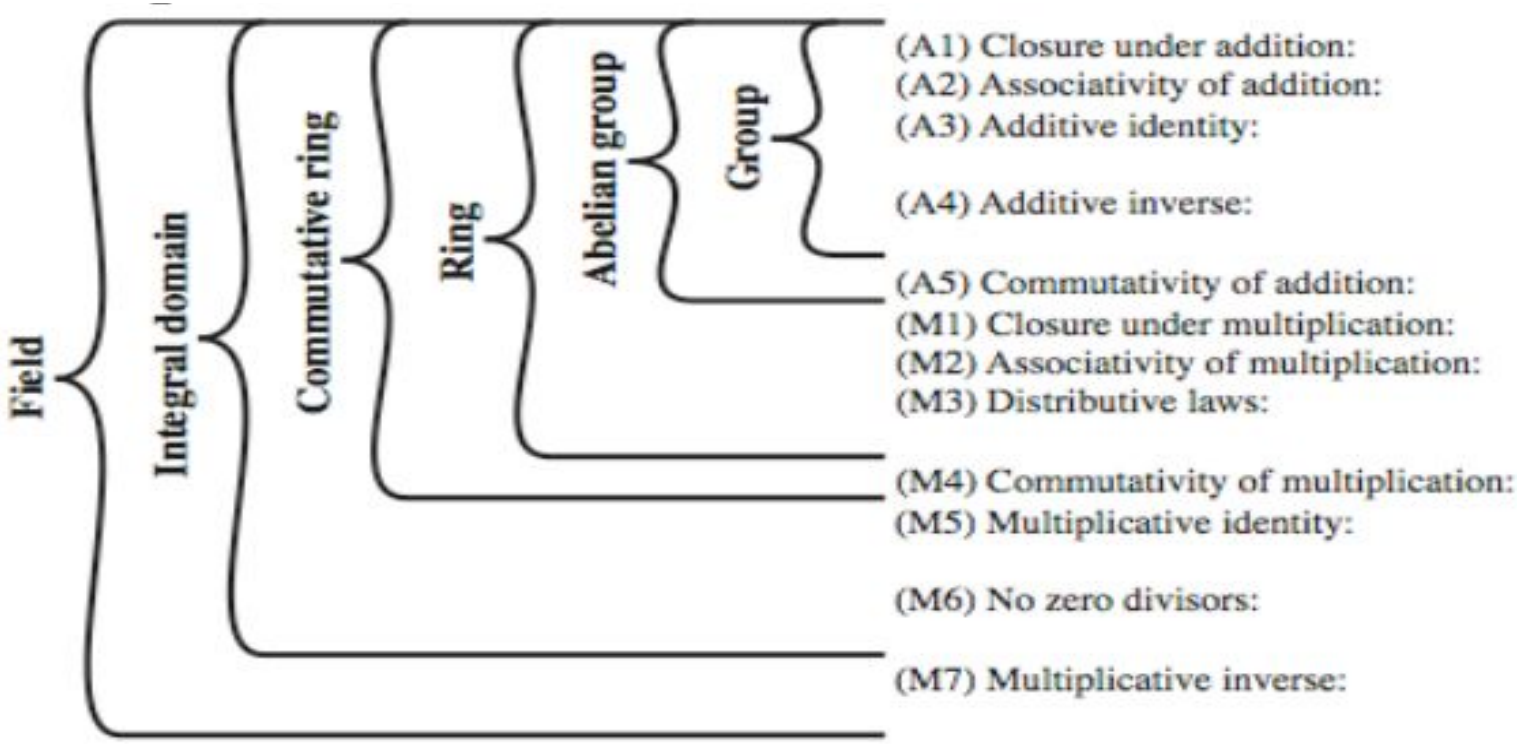
  - There is an element 1 such that a1 = 1a =a for all a R.
  - If $a, b \in R$ and $ab=0$, then either $a=0$ or $b=0$.

  - **Example:**

The set of integers (positive, negative and 0) under the usual operation of addition and multiplication.

Field

- An integral domain in which each element has a multiplicative inverse.



- The set of all real numbers under the operation of addition and multiplication is an example of field.
- In field we can do addition, subtraction, multiplication and division $\rightarrow$ $a/b = a(b^{-1})$

Finite Field or Galois Field

- **Finite Field**:
  - A field with finite number of elements.
  - Also known as Galois Field.
  - The number of elements is always a power (positive integer) of a prime number. Hence, denoted as ***GF(p^n)***
  - ***GF(p)*** is the set of integers, Z_p ={0,1, … , p-1} with arithmetic operations modulo prime p.
  - Can do addition, subtraction, multiplication, and division without leaving the field ***GF(p)***

Finite Field or Galois Field

- Any integer in Z_n has a multiplicative inverse if and only if that integer is relatively prime to n.
- There exists a multiplicative inverse for all of the nonzero integers in Z_p.
- Z_p is a field

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

- For Z_p the following equation holds.

$$\textbf{if } (a \times b) \equiv (a \times c) \pmod{p} \textbf{ then } b \equiv c \pmod{p}$$

When a is a relatively prime to p.

GF(2)

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.

GF(7)

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

Ordinary Polynomial Arithmetic

- A polynomial of degree n (n>=0) is an expression

  $f(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0 = \sum a_i x^i$

  where, S = a set of coefficients and $a_n \neq 0$
- Ordinary polynomial arithmetic includes addition, subtraction and multiplication.
- Division operation requires S to be a **field**.
- Example:

  let $f(x) = x^3 + x^2 + 2$ and $g(x) = x^2 - x + 1$

  $f(x) + g(x) = x^3 + 2x^2 - x + 3$

  $f(x) - g(x) = x^3 + x + 1$

  $f(x) \times g(x) = x^5 + 3x^2 - 2x + 2$

Ordinary Polynomial Arithmetic

$$
\begin{array}{r}
x^3 + x^2 \quad + 2 \\
+ \ (x^2 - x + 1) \\
\hline
x^3 + 2x^2 - x + 3
\end{array}
$$

(a) Addition

$$
\begin{array}{r}
x^3 + x^2 \quad + 2 \\
- \ (x^2 - x + 1) \\
\hline
x^3 \quad + x + 1
\end{array}
$$

(b) Subtraction

$$
\begin{array}{r}
x^3 + x^2 \quad + 2 \\
\times \ (x^2 - x + 1) \\
\hline
x^3 + x^2 \quad + 2 \\
- x^4 - x^3 \quad - 2x \\
x^5 + x^4 \quad + 2x^2 \\
\hline
x^5 \quad + 3x^2 - 2x + 2
\end{array}
$$

(c) Multiplication

$$
\begin{array}{r}
x + 2 \\
x^2 - x + 1 \,\big/\, x^3 + x^2 \quad + 2 \\
x^3 - x^2 + x \\
\hline
2x^2 - x + 2 \\
2x^2 - 2x + 2 \\
\hline
x
\end{array}
$$

(d) Division

Polynomial Arithmetic with Coefficients in Z_p

- When polynomial arithmetic is performed on polynomials over a field, then division is possible.
- Example:
  - If coefficients are integers, then $(5x^2/3x) \rightarrow$ does not have solution.
  - If coefficients are from Z_7, then $(5x^2/3x) \rightarrow 4x$.
- Given polynomials $f(x)$ of degree n and $g(x)$ of m $(n>=m)$, we can write $f(x) = q(x)g(x)+r(x)$

  where

  $deg(f(x))=n$

  $deg(g(x))=m$

  $deg(q(x))=n-m$

  $deg(r(x))<=m-1$

Polynomial Arithmetic with Coefficients in Z_p

- A polynomial ***f(x)*** over a field ***F*** is 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)***.
- Example: [***F=GF(2)***]

  $f(x) = x^3+x+1$ is a irreducible polynomial.

Polynomial Arithmetic over GF(2)

$$
\begin{array}{rrrrrrrr}
 & x^7 & +x^5 & +x^4 & +x^3 & & +x+1 \\
 & & & & +(x^3 & & +x+1) \\
\hline
 & x^7 & +x^5 & +x^4 & & & &
\end{array}
$$

(a) Addition

$$
\begin{array}{rrrrrrrr}
 & x^7 & +x^5 & +x^4 & +x^3 & & +x+1 \\
 & & & & -(x^3 & & +x+1) \\
\hline
 & x^7 & +x^5 & +x^4 & & & &
\end{array}
$$

(b) Subtraction

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

(c) Multiplication

$$
\begin{array}{rl}
 & x^4+1 \\
x^3+x+1 \;\big/ & \begin{array}{rrrrrrr}
x^7 & +x^5 & +x^4 & +x^3 & & +x+1 \\
x^7 & +x^5 & +x^4 & & & & \\
\hline
 & & & x^3 & & +x+1 \\
 & & & x^3 & & +x+1 \\
\hline
\end{array}
\end{array}
$$

Polynomial GCD

- Polynomial c(x) is the greatest common divisor of a(x) and b(x) if :
  - c(x) divides both a(x) and b(x)
  - Any divisor of a(x) and b(x) is a divisor of c(x)
  - c(x) is the polynomial of maximum degree that divides both a(x) and b(x).

Find $\gcd[a(x), b(x)]$ for $a(x) = x^6 + x^5 + x^4 + x^3 + x^2 + x + 1$ and $b(x) = x^4 + x^2 + x + 1$. First, we divide $a(x)$ by $b(x)$:

$$
\begin{array}{r}
x^2 + x \\
x^4 + x^2 + x + 1 \overline{)\, x^6 + x^5 + x^4 + x^3 + x^2 + x + 1} \\
\underline{x^6 \qquad + x^4 + x^3 + x^2} \\
x^5 \qquad\qquad\quad + x + 1 \\
\underline{x^5 \qquad + x^3 + x^2 + x} \\
x^3 + x^2 \qquad + 1
\end{array}
$$

This yields $r_1(x) = x^3 + x^2 + 1$ and $q_1(x) = x^2 + x$.
Then, we divide $b(x)$ by $r_1(x)$.

$$
\begin{array}{r}
x + 1 \\
x^3 + x^2 + 1 \overline{)\, x^4 \qquad + x^2 + x + 1} \\
\underline{x^4 + x^3 \qquad\quad + x} \\
x^3 + x^2 \qquad + 1 \\
\underline{x^3 + x^2 \qquad + 1}
\end{array}
$$

This yields $r_2(x) = 0$ and $q_2(x) = x + 1$.
Therefore, $\gcd[a(x), b(x)] = r_1(x) = x^3 + x^2 + 1$.

Motivation for Finite Field of Form $GF(2^n)$

- All encryption algorithm requires arithmetic operations on integers.
- Need integers in the range ***0*** through 2^n-1, which fit into an ***n***-bit word with no wasted bit patterns.

> Suppose we wish to define a conventional encryption algorithm that operates on data 8 bits at a time, and we wish to perform division. With 8 bits, we can represent integers in the range 0 through 255. However, 256 is not a prime number, so that if arithmetic is performed in Z_{256} (arithmetic modulo 256), this set of integers will not be a field. The closest prime number less than 256 is 251. Thus, the set Z_{251}, using arithmetic modulo 251, is a field. However, in this case the 8-bit patterns representing the integers 251 through 255 would not be used, resulting in inefficient use of storage.

- The set of integers modulo 2^n is not a field. Even if only addition and multiplication are required, the use of Z_{2n} is not good choice.

Motivation for Finite Field of Form $GF(2^n)$

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

Integer	1	2	3	4	5	6	7
Occurrences in Z_8	4	8	4	12	4	8	4
Occurrences in $GF(2^3)$	7	7	7	7	7	7	7

Modular Polynomial Arithmetic

- ***S*** = set of all polynomials of degree n-1 or less over $\boldsymbol{Z_p}$ where coefficients are taken from {0,1,….,p-1}. There are a total of $\boldsymbol{p^n}$ polynomials in ***S***.

For $p = 3$ and $n = 2$, the $3^2 = 9$ polynomials in the set are

0	x	$2x$
1	$x + 1$	$2x + 1$
2	$x + 2$	$2x + 2$

For $p = 2$ and $n = 3$, the $2^3 = 8$ polynomials in the set are

0	$x + 1$	$x^2 + x$
1	x^2	$x^2 + x + 1$
x	$x^2 + 1$	

- ***S*** is a finite field, where
  - Arithmetic follows the ordinary rules of polynomial arithmetic.
  - Arithmetic on coefficients is performed modulo p.
  - If a polynomial of degree greater than n-1 is generated from multiplication, then the polynomial is reduced modulo some irreducible m(x) of degree n.

Modular Polynomial Arithmetic

The Advanced Encryption Standard (AES) uses arithmetic in the finite field GF(2^8), with the irreducible polynomial $m(x) = x^8 + x^4 + x^3 + x + 1$. Consider the two polynomials $f(x) = x^6 + x^4 + x^2 + x + 1$ and $g(x) = x^7 + x + 1$. Then

$$\begin{aligned} f(x) + g(x) &= x^6 + x^4 + x^2 + x + 1 + x^7 + x + 1 \\ &= x^7 + x^6 + x^4 + x^2 \end{aligned}$$

$$\begin{aligned} f(x) \times g(x) = {} & x^{13} + x^{11} + x^9 + x^8 + x^7 \\ & + x^7 + x^5 + x^3 + x^2 + x \\ & + x^6 + x^4 + x^2 + x + 1 \\ = {} & x^{13} + x^{11} + x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + 1 \end{aligned}$$

$$\begin{array}{r} x^5 + x^3 \\ x^8 + x^4 + x^3 + x + 1 \overline{)\, x^{13} + x^{11} + x^9 + x^8 + x^7 + x^6 + x^5 + x^4 + x^3 + 1} \\ \underline{x^{13} \qquad + x^9 + x^8 \qquad + x^6 + x^5} \\ x^{11} \qquad\qquad\qquad + x^4 + x^3 \\ \underline{x^{11} \qquad\quad + x^7 + x^6 \qquad + x^4 + x^3} \\ x^7 + x^6 \qquad\qquad\qquad\quad + 1 \end{array}$$

Therefore, $f(x) \times g(x) \bmod m(x) = x^7 + x^6 + 1$.

Modular Polynomial Arithmetic

- The set of residues modulo ***m(x)***, an ***n***th degree polynomial, consists of p^n elements where each of the elements is represented by one of the p^n polynomials of degree $m<n = S$.
- Example GF(2^3)

Table 4.6 Polynomial Arithmetic Modulo ($x^3 + x + 1$)

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

(a) Addition

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

(b) Multiplication

Computational Considerations

- since coefficients are 0 or 1, can represent any such polynomial as a bit string
- addition becomes XOR of these bit strings
- multiplication is shift and XOR
- modulo reduction done by repeatedly substituting highest power with remainder of irreducible poly (also shift and XOR)
- eg. irreducible poly = $x^3 + x + 1$ **means** $x^3 = x + 1$ in the polynomial field

Computational Example

- in GF(2^3) have (x^2+1) is 101_2 & (x^2+x+1) is 111_2
- so addition is
  - $(x^2+1) + (x^2+x+1) = x$
  - $101 \text{ XOR } 111 = 010_2$
- and multiplication is
  - $(x+1).(x^2+1) = x.(x^2+1) + 1.(x^2+1)$
    $= x^3+x + x^2+1 = x^3+x^2+x+1$
  - $011.101 = (101)<<1 \text{ XOR } (101)<<0 =$
    $1010 \text{ XOR } 0101 = 1111_2$
- polynomial modulo reduction (to get q(x) & r(x))
  - $(x^3+x^2+x+1) \bmod (x^3+x+1) = 1.(x^3+x+1) + (x^2) = x^2$
  - $1111 \bmod 1011 = 1111 \text{ XOR } 1011 = 0100_2$

- Basic concepts in number theory and finite fields from the book of William Stallings (Chapter 2 and Chapter 5).
- Also refer to various youtube clips on the discussed topic.

Cryptography and Security

Lecture 5

Pseudorandom Number Generation and Introduction to Stream Cipher

Lecture slides are adopted from slides of Dan Boneh

Review

- Cipher over (K,M,C): a pair of "efficient" algs (***E***, ***D***) s.t. $\forall\ m \in M, k \in K$: ***D***(k, ***E***(k, m)) = m

- A good cipher: **OTP** $M=C=K=\{0,1\}^n$

  $E(k, m) = k \oplus m$, $D(k, c) = k \oplus c$

- **Lemma: OTP** has perfect secrecy (i.e. no CT only attacks)

- **Bad news:** perfect-secrecy $\Rightarrow$ key-len $\geq$ msg-len

Stream Ciphers: making OTP practical

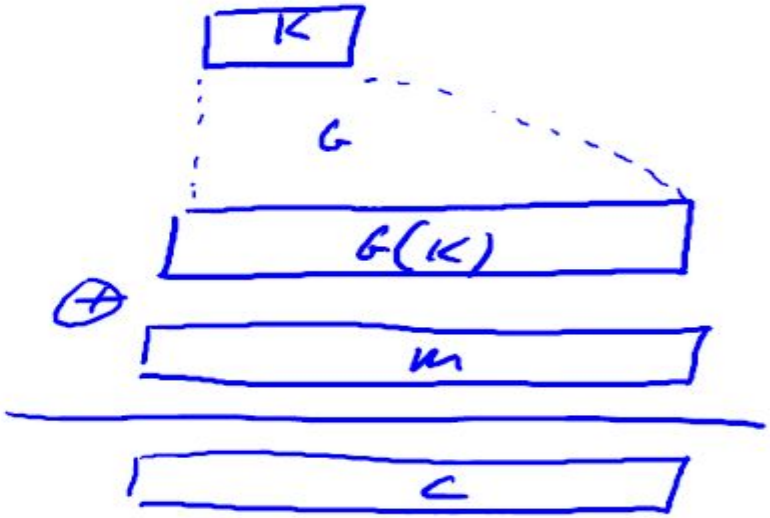
- idea: replace "random" key by "pseudorandom" key.

PRG is a function $G: \underbrace{\{0,1\}^s}_{\text{seed space}} \rightarrow \{0,1\}^n$ $\quad n \gg s$

(eff. computable by a deterministic algorithm)

$C := E(K,m) = m \oplus G(K)$

$D(K,C) = C \oplus G(K)$



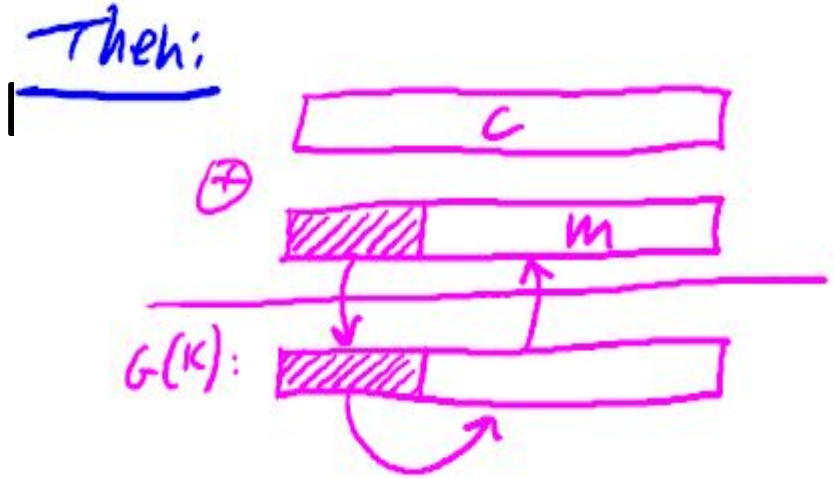
Stream Ciphers: making OTP practical

- Stream ciphers cannot have perfect secrecy.
- Need a new definition of perfect secrecy based on specific secrecy.

PRG must be unpredictable

- Suppose PRG is predictable:

  $\exists i: G(K)|1,\ldots,_i \rightarrow G(K)|_{i+1,\ldots,n}$



- Even $G(K)|1,\ldots,_i \rightarrow G(K)|_{i+1}$

PRG must be unpredictable

- G: $K \longrightarrow \{0,1\}^n$ is **predictable** if:

$\exists$ "eff" alg. A and $\exists\, 0 \le i \le n-1$ s.t.

$$\Pr_{k \xleftarrow{R} K}\left[A(G(k))\big|_{1,\ldots,i} = G(k)\big|_{i+1} \right] > \frac{1}{2} + \varepsilon$$

For non-negligible ε (e.g. $\varepsilon = 1/2^{30}$)

- Def: PRG is **unpredictable** if it is not predictable

$\Rightarrow \forall$ i: no "eff" adv. can predict bit (i+1) for "non-neg" ε

PRG Security Definition

- Let $G:K \longrightarrow \{0,1\}^n$ be a PRG
- Goal: We need to show that

$[k \xleftarrow{R} K,\ \text{output } G(k)]$

is "indistinguishable" from

$[r \xleftarrow{R} \{0,1\}^n,\ \text{output } r]$

Statistical Tests

- **Statistical test** on $\{0,1\}^n$:

  An algorithm A that outputs A(x)=0 when x is not random or outputs A(x)=1 when x is random.

Examples:

(1) $A(x)=1$ iff $\left| \#0(x) - \#1(x) \right| \leq 10 \cdot \sqrt{n}$

(2) $A(x)=1$ iff $\left| \#00(x) - \frac{n}{4} \right| \leq 10 \cdot \sqrt{n}$

(3) $A(x)=1$ iff max-run-of-0$(x) < 10 \cdot \log_2(n)$

Advantage of Statistical Test

- Let $G:K \longrightarrow \{0,1\}^n$ be a PRG and A a stat. test on $\{0,1\}^n$. The advantage of A with respect to G is

$$Adv_{PRG}[A,G] = \left| \Pr_{K \xleftarrow{R} K}[A(G(K))=1] - \Pr_{r \xleftarrow{R} \{0,1\}^n}[A(r)=1] \right| \in [0,1]$$

- If Adv is close to 1 → A can distinguish G(k) from r.
- If Adv is close to 0 → A cannot distinguish G(k) from r.
- Example: A(x)=0 then $Adv_{PRG}[A,G]=0$

Advantage of Statistical Test

- **Example:**

  Suppose $G:K \longrightarrow \{0,1\}^n$ satisfies **msb(G(k)) = 1** for 2/3 of keys in K

- Define stat. test A(x) as:

  if [msb(x)=1] then output "1" else output "0"

- Adv_{PRG} [A,G] = | Pr[A(G(k))=1] - Pr[A(r)=1] | = 2/3 - ½ = 1/6.

Secure PRG

- G:K $\longrightarrow \{0,1\}^n$ is a **secure PRG** if

$\forall$ "eff" std. tests A:

$Adv_{PRG}[A, G]$ is "negligible"

Secure PRG

- Easy fact: **a secure PRG is unpredictable**
- We show: PRG predictable ⇒ PRG is insecure
- Suppose A is an efficient algorithm s.t.

$$\Pr_{k \xleftarrow{R} \{0,1\}^k}\left[A(G(k)|_{1,\dots,i}) = G(k)|_{i+1} \right] > \frac{1}{2} + \varepsilon$$

- Define statistical test B as:

$$B(x) = \begin{cases} \text{if } A(x|_{1,\dots,i}) = x_{i+1} \text{ output } 1 \\ \text{else output } 0 \end{cases}$$

$r \xleftarrow{R} \{0,1\}^n$: $\Pr[B(r)=1] = \frac{1}{2}$

$k \xleftarrow{R} \{0,1\}^k$: $\Pr[B(G(k))=1] > \frac{1}{2} + \varepsilon$

$\Rightarrow Adv_{PRG}[B,G] = \left| \Pr[B(r)=1] - \Pr[B(G(k))=1] \right| > \varepsilon$

Secure PRG

- **Thm (Yao'82): an unpredictable PRG is secure**

- Let $G:K \longrightarrow \{0,1\}^n$ be PRG

- "Thm": if $\forall$ i $\in$,0, … , n-1} PRG G is unpredictable at pos. i then G is a secure PRG.

Computational Indistinguishability of Two Distributions

- Let P_1 and P_2 be two distributions over $\{0,1\}^n$
- Def: We say that P_1 and P_2 are **computationally indistinguishable** (denoted $P_1 \approx_p P_2$)

if $\forall$ "eff" stat. tests A

- $\left| \Pr_{x \leftarrow P_1}[A(x)=1] - \Pr_{x \leftarrow P_2}[A(x)=1] \right| < \text{negligible}$

a PRG is secure if $\{ k \leftarrow K : G(k) \} \approx_p \text{uniform}(\{0,1\}^n)$

Semantic Security

- Goal: secure PRG ⇒ “secure” stream cipher
- Attacker’s abilities: **obtains one ciphertext**
- Shannon’s idea:

  CT should reveal no “info” about PT

Shannon's Perfect Secrecy

Let (E,D) be a cipher over (K,M,C)

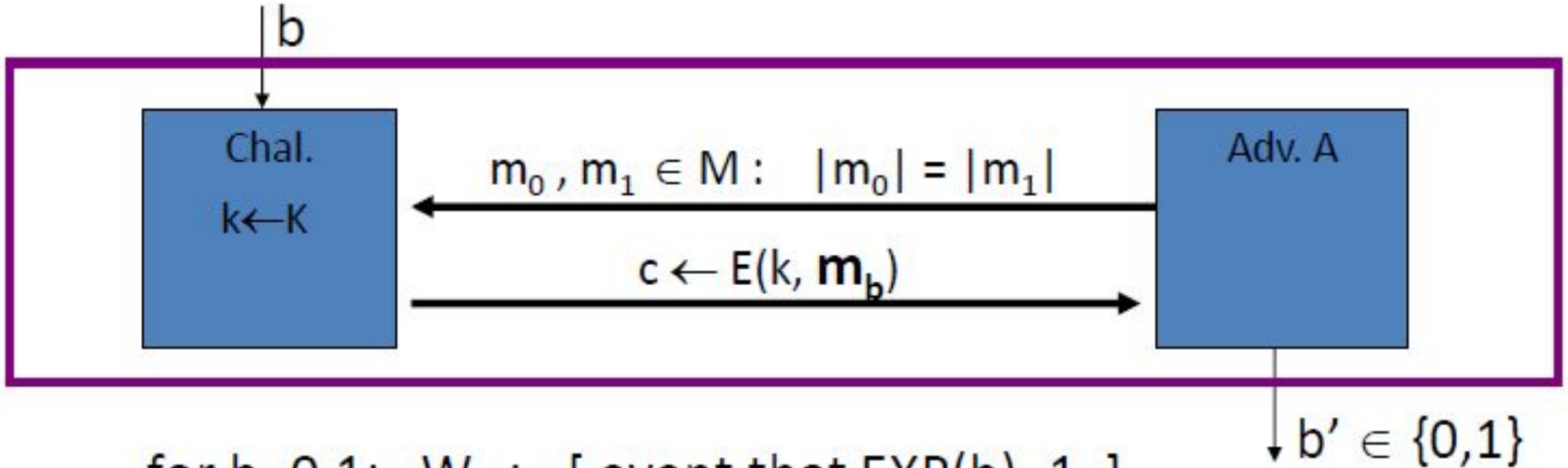
- (E,D) has perfect secrecy if $\forall\ m_0, m_1 \in M\ (\ |m_0| = |m1|\ )$
  $\{\ E(k,m_0)\ \} = \{\ E(k,m_1)\ \}$ where $k \leftarrow\!\!- K$

- (E,D) has perfect secrecy if $\forall\ m_0, m_1 \in M\ (\ |m_0| = |m_1|\ )$
  $\{\ E(k,m_0)\ \} \approx_{\mathbf{p}} \{\ E(k,m_1)\ \}$ where $k \leftarrow\!\!- K$

- … but also need adversary to exhibit $m_0, m_1 \in M$ explicitly

Semantic Security (one-time key)

For b=0,1 define experiments EXP(0) and EXP(1) as:



for b=0,1: W_b := [event that EXP(b)=1]

$$Adv_{SS}[A,E] := \left| \Pr[W_0] - \Pr[W_1] \right| \in [0,1]$$

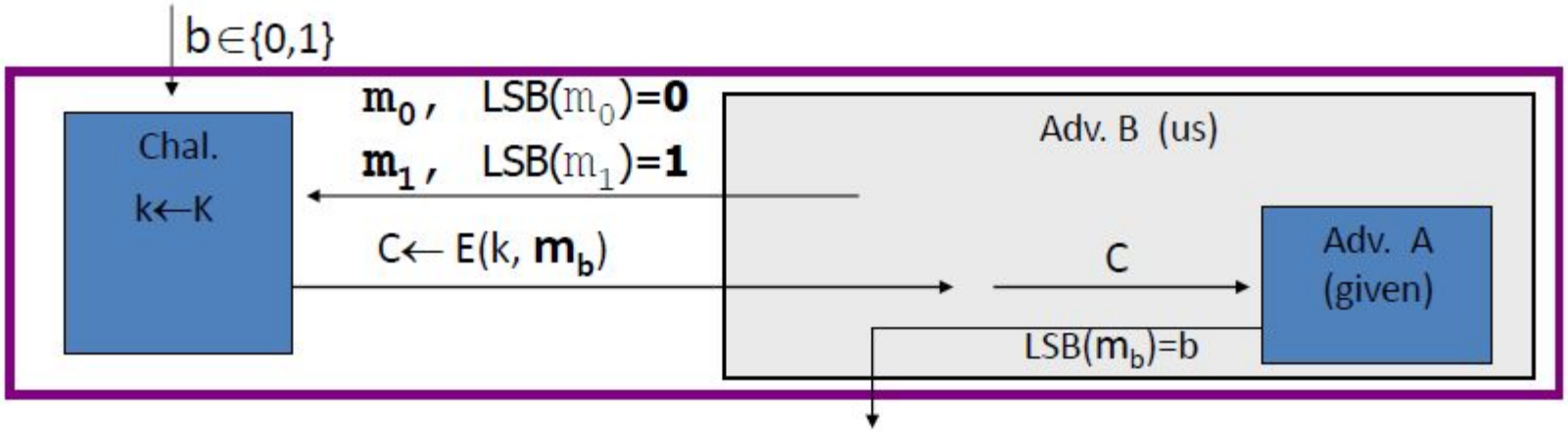
Semantic Security (one-time key)

- E is **semantically secure** if for all efficient A $Adv_{SS}[A,E]$ is negligible.
- $\Rightarrow$ for all explicit $m_0 , m_1 \in M$:

  $\{ E(k,m_0) \} \approx_{\mathbf{p}} \{ E(k,m_1) \}$

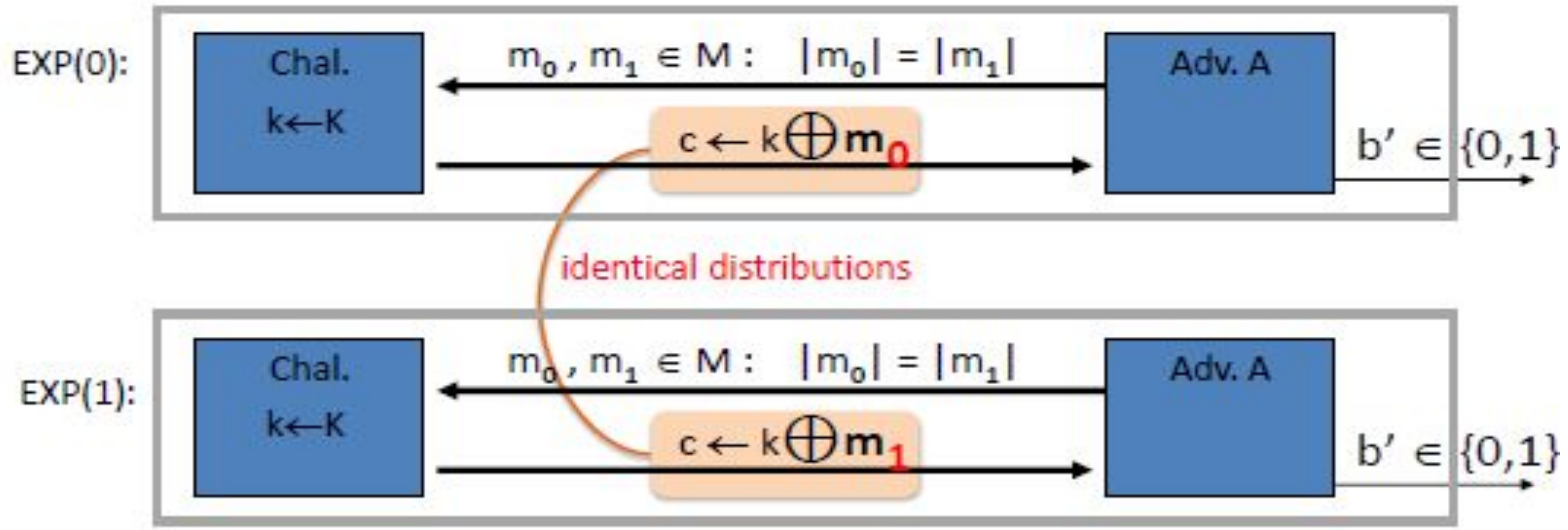
Example

- Suppose efficient A can always deduce LSB of PT from CT.
- ⇒ E = (E,D) is not semantically secure.



$$Adv_{SS}[B, E] = \left| Pr[\,EXP(0)=1\,] - Pr[\,EXP(1)=1\,] \right| = 0-1=1$$

OTP is semantically secure



For **all** A: $Adv_{SS}[A,OTP] = \left| Pr[A(k \oplus m_0)=1] - Pr[A(k \oplus m_1)=1] \right|$ **=0**

Stream ciphers are semantically secure

- **Goal:**

  secure PRG ⇒ semantically secure stream cipher

- **Thm:**

  G:K $\longrightarrow\{0,1\}^n$ is a secure PRG ⇒ stream cipher E derived from G is sem. sec.

Prove by Contrapositive:

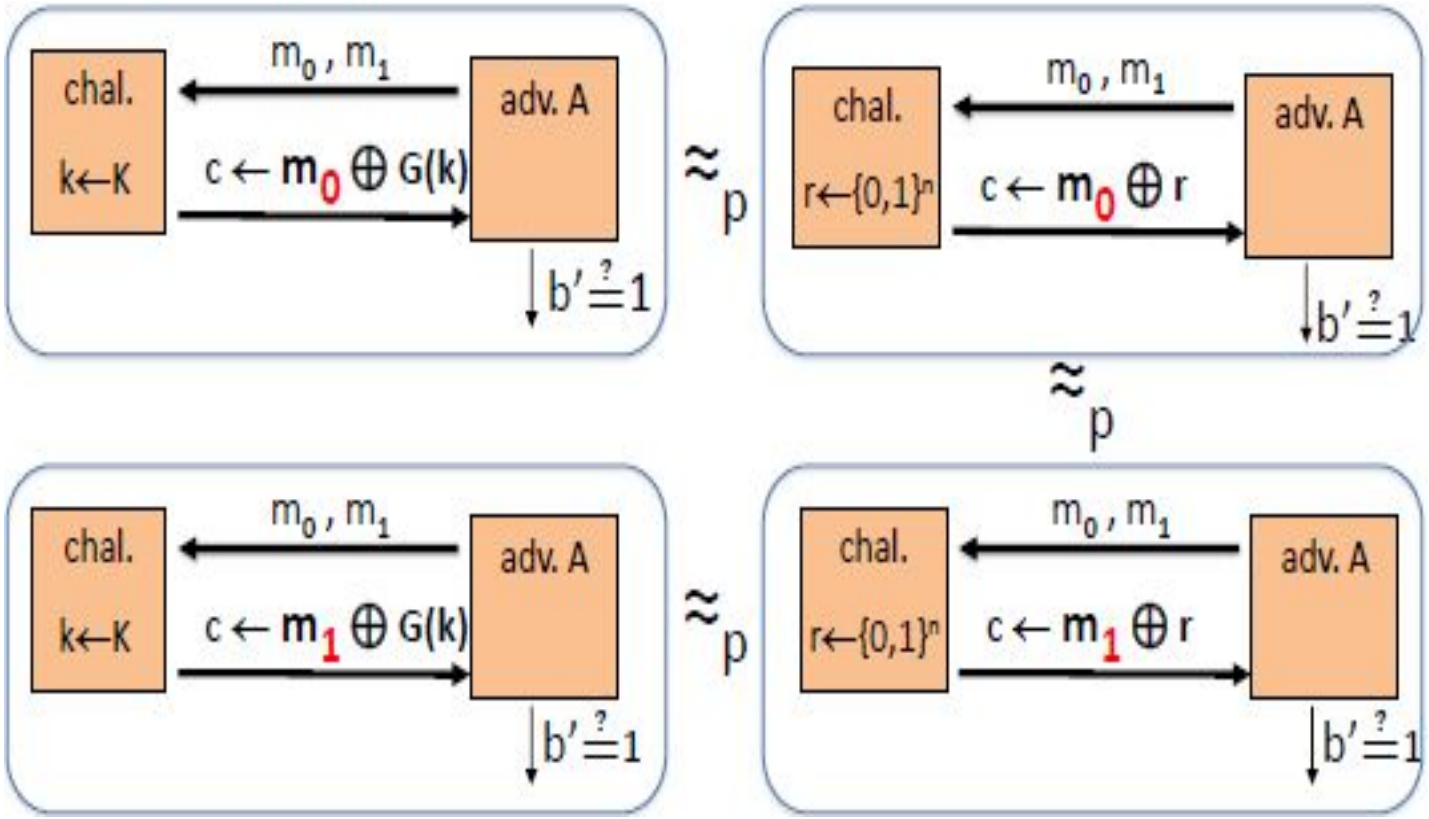
  Stream cipher is insecure ⟶ PRG used in not secure

∀ sem. sec. adversary A , ∃ a PRG adversary B s.t.

$$\mathbf{Adv_{SS}[A,E] \leq 2 \cdot Adv_{PRG}[B,G]}$$

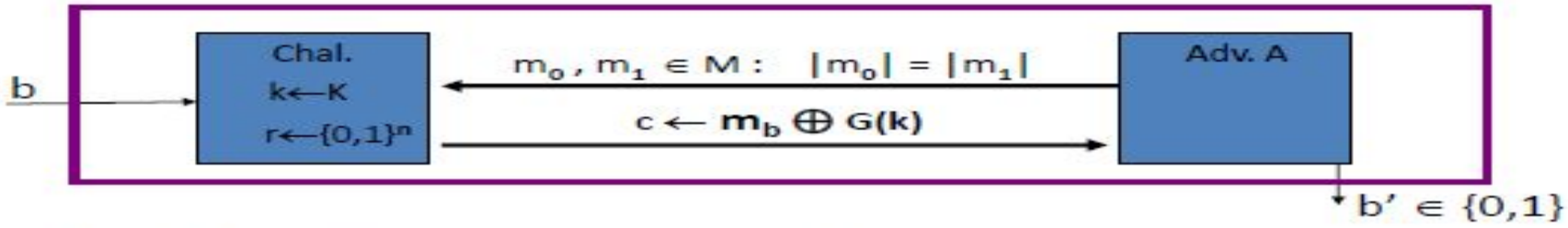
Intuition of Proof

Let A be a semantic security adversary.



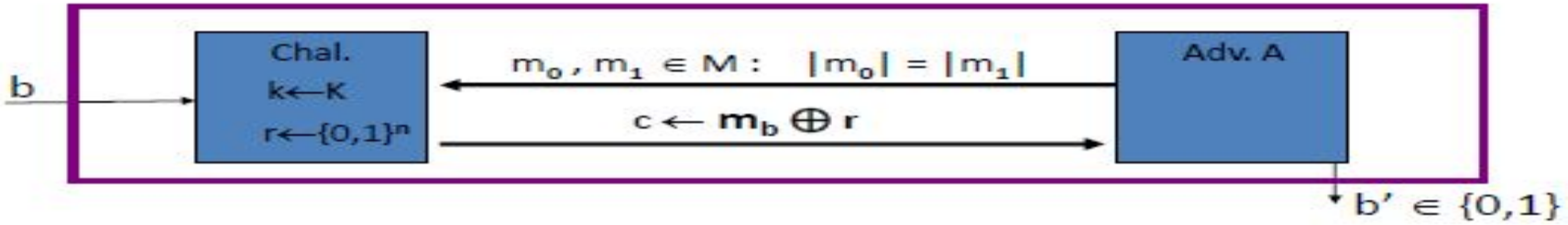
Stream ciphers are semantically secure

Proof: Let A be a sem. sec. adversary.



For b=0,1: W_b := [event that b'=1].

$$Adv_{SS}[A,E] = \left| Pr[W_0] - Pr[W_1] \right|$$



For b=0,1: W_b := [event that b'=1].

$$Adv_{SS}[A,E] = \left| Pr[W_0] - Pr[W_1] \right|$$

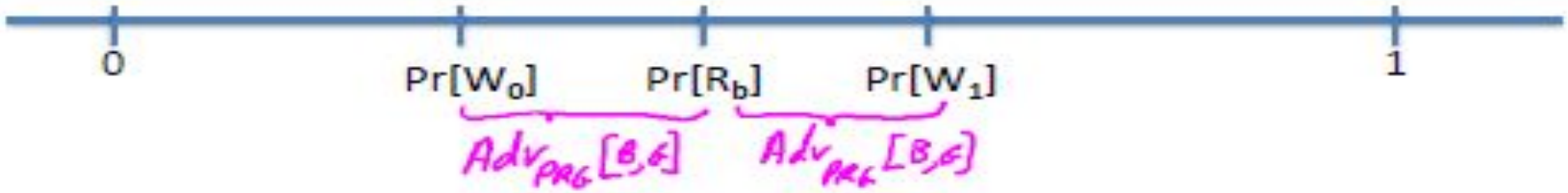
For b=0,1: R_b := [event that b'=1]

Stream ciphers are semantically secure

Proof: Let A be a sem. sec. adversary.

Claim 1: $|\Pr[R_0] - \Pr[R_1]| = Adv_{SS}[A, OTP] = 0$

Claim 2: $\exists B: |\Pr[W_b] - \Pr[R_b]| = Adv_{PRG}[B, G]$ For $b=0,1$

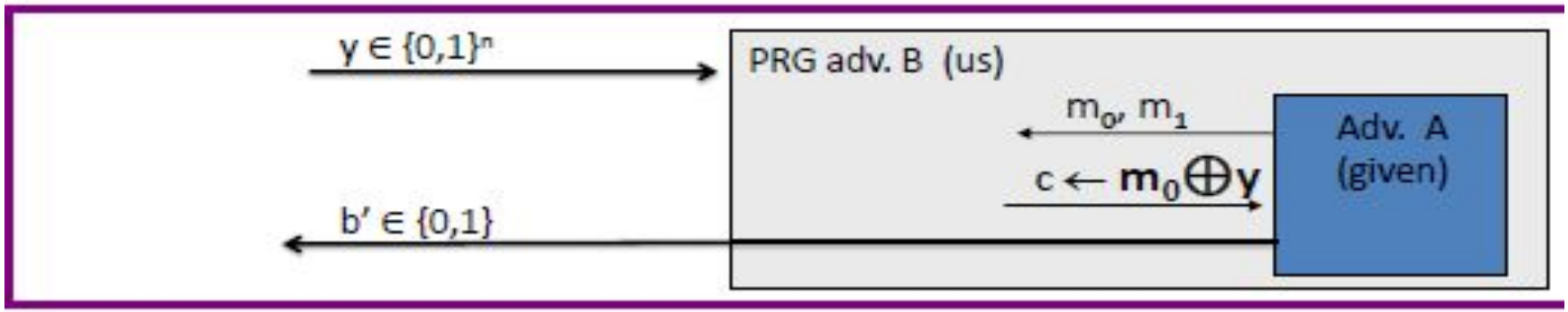


$$\Rightarrow \mathrm{Adv}_{SS}[A,E] = \left|\Pr[W_0] - \Pr[W_1]\right| \leq 2 \cdot \mathrm{Adv}_{PRG}[B,G]$$

Stream ciphers are semantically secure

Proof of claim 2: $\exists B: \left|\Pr[W_0] - \Pr[R_0]\right| = \mathrm{Adv}_{PRG}[B,G]$

Algorithm B:



$$\mathrm{Adv}_{PRG}[B,G] = \left| \Pr_{r \xleftarrow{R} \{0,1\}^n}[B(r)=1] - \Pr_{k \xleftarrow{R} \mathcal{K}}[B(G(k))=1] \right| = \left| \Pr[R_0] - \Pr[W_0] \right|$$

Cryptography and Security

Lecture 6

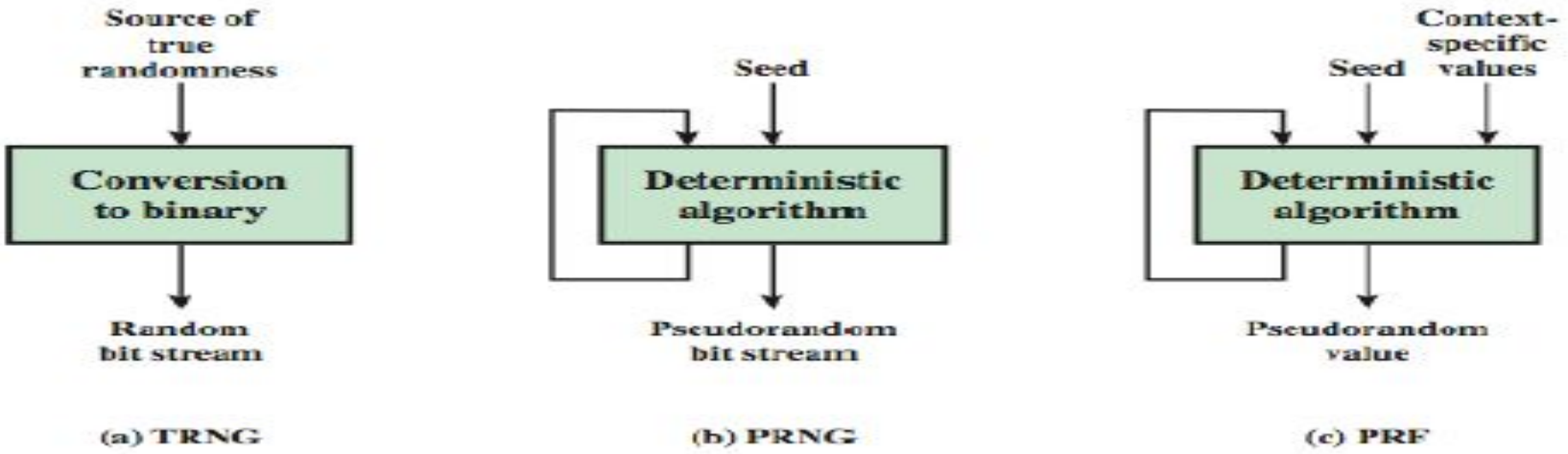
Pseudorandom Number Generation and Introduction to Stream Cipher

Random Number

- Many uses of **random numbers** in cryptography
  – Nonces in authentication protocols to prevent replay attack.
  – session keys
  – public key generation
  – keystream for a one-time pad
- Criteria of Random Number:
  – Uniform Distribution.
  – Independence
  – Unpredictability
- Care should be taken during the generation of pseudo random number.

(Pseudo) Random Number Generators

- Often use deterministic algorithmic techniques to create "random numbers"
  - not truly random
  - can pass many tests of "randomness"
- known as "pseudorandom numbers"
- created by "Pseudorandom Number Generators (PRNGs)"



(Pseudo) Random Number Generators

- **Entropy Source:**
  - Source of true randomness.
  - Drawn from the physical environment of the computer such as keystroke timing patterns, disk electrical activity, mouse movements and instantaneous values of the system clock.
- **Seed:** Often generated by TRNG.
- If the adversary knows the algorithm and the seed, then s/he can reproduce the entire bit stream.

PRNG Requirements

- **Randomness**
  - uniformity, scalability, consistency
- **Unpredictability**
  - forward & backward unpredictability
- **Characteristics of the seed**
  - secure
  - if known adversary can determine output
  - so must be random or pseudorandom number

Linear Congruential Generator

- Common iterative technique using:

$$X_{n+1} = (aX_n + c) \bmod m$$

- Given suitable values of parameters can produce a long random-like sequence
- Suitable criteria to have are:
  - function generates a full period
  - generated sequence should appear random
  - efficient implementation with 32-bit arithmetic
- Note that an attacker can reconstruct sequence given a small number of values (knowing *a, c and m*)

Blum Blum Shub Generator

- Based on public key algorithms
- Use least significant bit from iterative equation:

$$X_0 = s^2 \bmod n$$
$$\textbf{for } i = 1 \textbf{ to } \infty$$
$$X_i = (X_{i-1})^2 \bmod n$$
$$B_i = X_i \bmod 2$$

- Unpredictable, passes **next-bit** test
- security rests on difficulty of factoring n
- is unpredictable given any run of bits
- slow, since very large numbers must be used
- too slow for cipher use, good for key generation

Using Block Ciphers as PRNGs

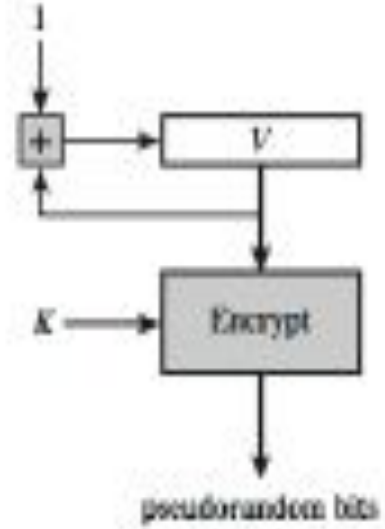
- For cryptographic applications, can use a block cipher to generate random numbers

- CTR

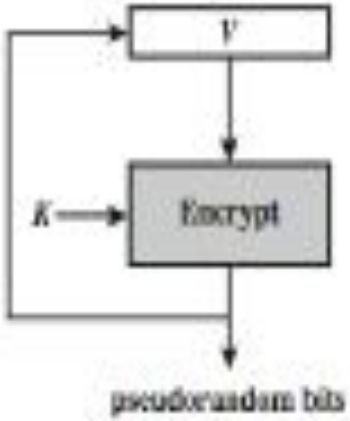
$$X_i = \mathrm{E}_K[\mathrm{V}_i]$$

- OFB

$$X_i = \mathrm{E}_K[X_{i-1}]$$



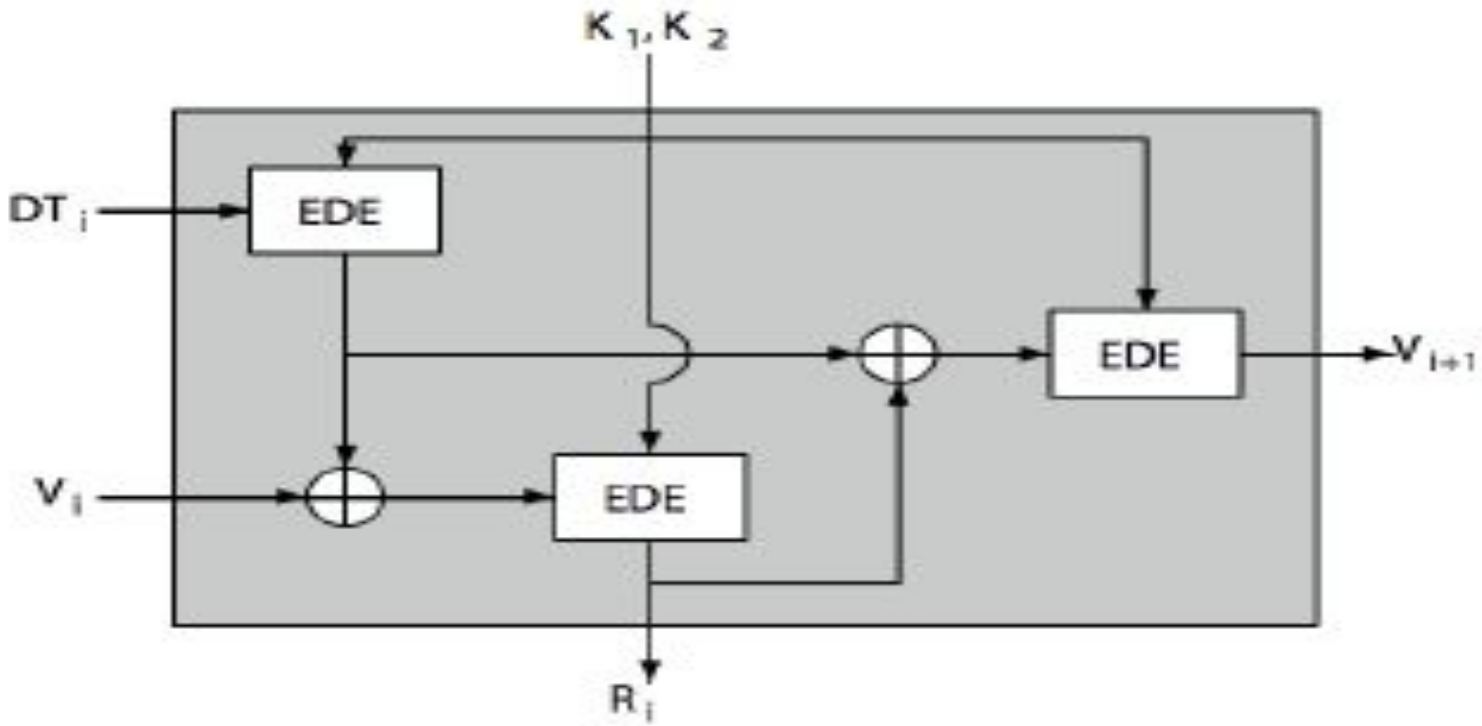
(a) CTR Mode



(b) OFB Mode

ANSI X9.17 PRNG

- A relatively complicated construction, including timestamps, and multiple encryptions
- 64 bit date and time (DT_i) and 64 bit seed value
- Uses 2-key (K_1,K_2) triple DES



Stream Cipher

- process message bit by bit (as a stream)
  - have a pseudo random **keystream**
- combined (XOR) with plaintext bit by bit
  - $C_i = M_i \; XOR \; StreamKey_i$

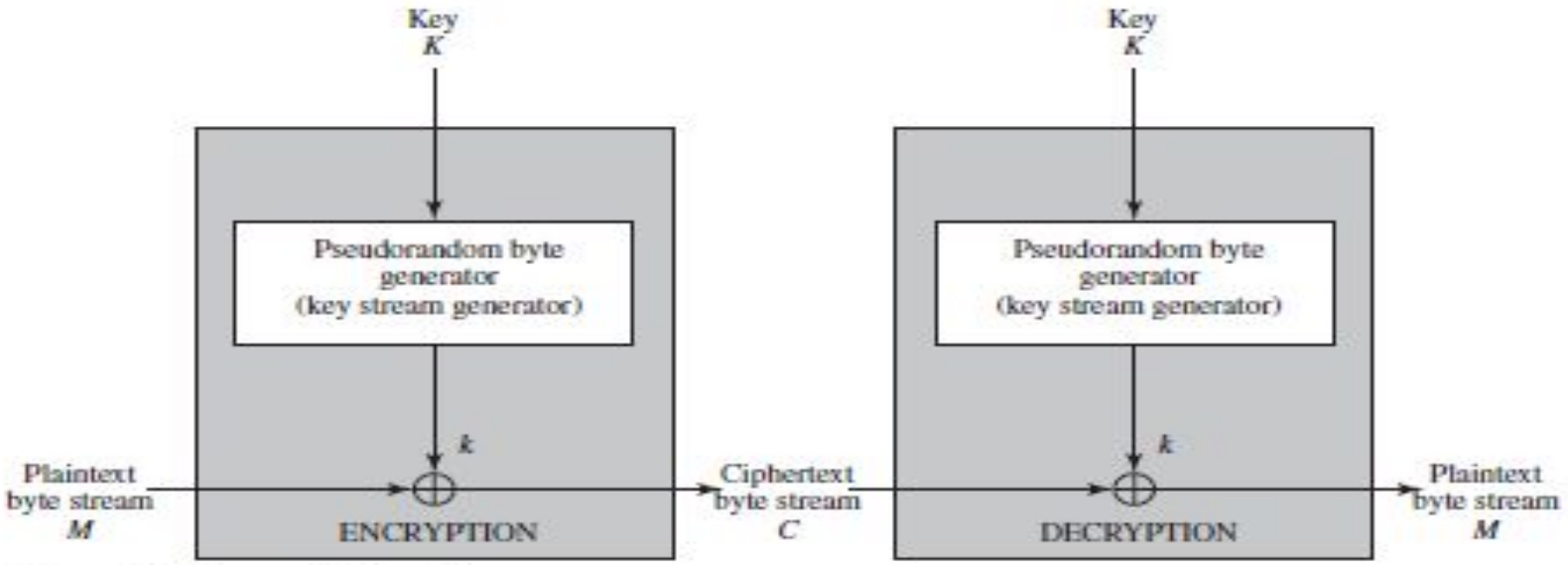


Figure 7.5 Stream Cipher Diagram

Stream Cipher

- Design Consideration for Stream Cipher:
  - The encryption period should have a large period.
  - Keystream should approximate the properties of a true random number stream as close as possible.
  - Key to generate the pseudorandom should be sufficiently large to defend the brute-force attack.
- Never reuse stream key
  - otherwise can recover messages (via XOR of msgs)
- Stream cipher can be used in data communication channel or a browser/web link.

RC4

- a proprietary cipher owned by RSA security.
- Another Ron Rivest design, simple but effective
- Variable key size, byte-oriented stream cipher
- Widely used (web SSL/TLS, wireless WEP/WPA)

RC4 Key Schedule

- Starts with an array **S** of numbers: 0....255
- Use key to well and truly shuffle **S**
- **S** forms **internal state** of the cipher

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

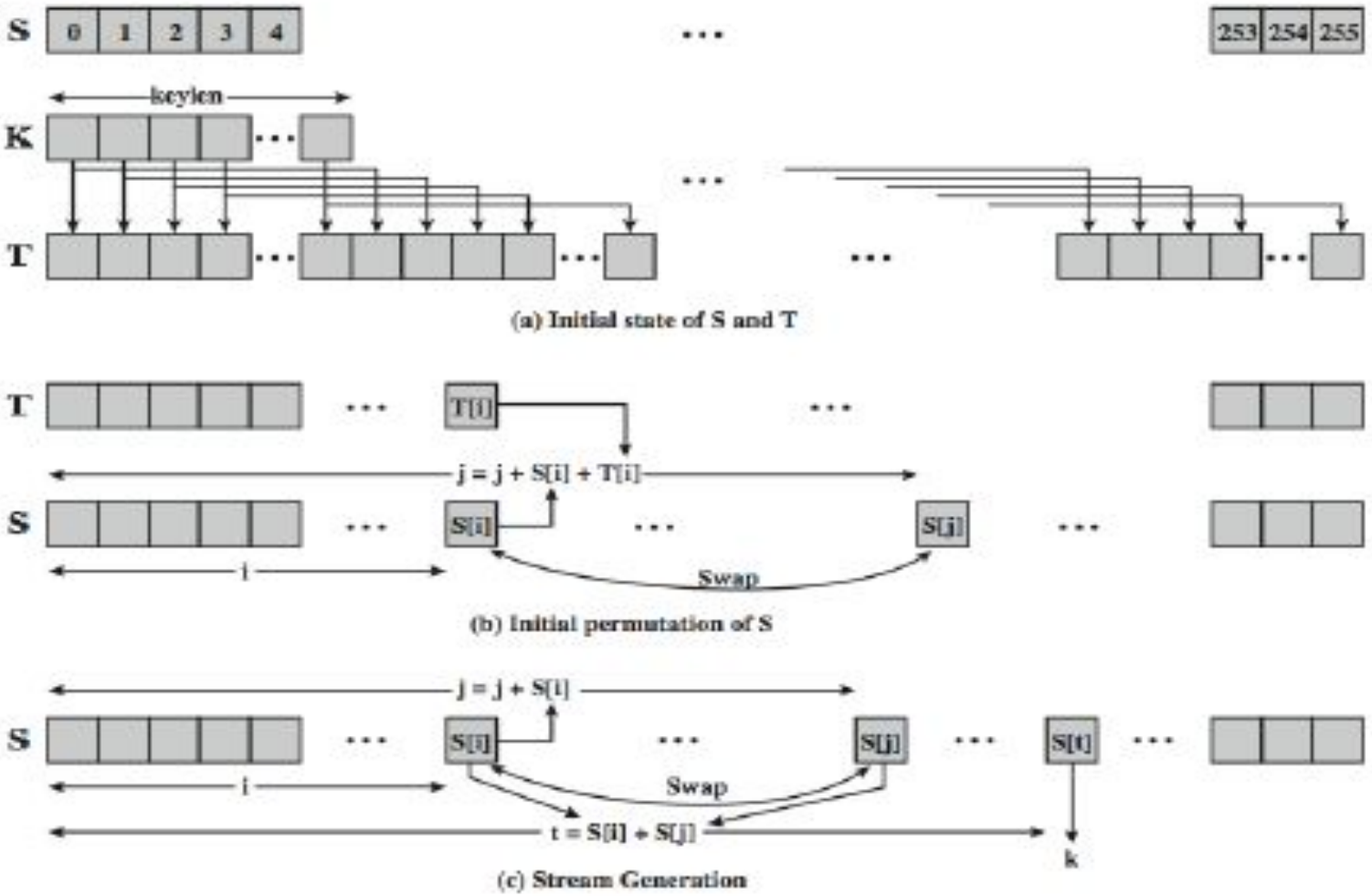
RC4 Encryption

- Encryption continues shuffling array values
- Sum of shuffled pair selects "stream key" value from permutation
- XOR **S[t]** with next byte of message to en/decrypt

```
for each message byte M_i
    i = (i + 1) (mod 256)
    j = (j + S[i]) (mod 256)
    swap (S[i], S[j])
    t = (S[i] + S[j]) (mod 256)
    C_i = M_i XOR S[t]
```

- used in the WiFi Protected Access (WPA) .It is optional for use in Secure Shell (SSH) and Kerberos.

RC4 Overview



True Random Number Generator

- **Entropy Source:**
  - Non-deterministic source to generate randomness.
  - Measure unpredictable natural processes such as pulse detector of radiation event, gas discharge tubes and leaky capacitor.
  - Generators are commercially available.
  - Some sources for computers:
    - Sound / video input.
    - Disk drive.
    - **Skew:**
  - Output from TRNG may contains some bias.
  - Deskewing process:
    - » Pass though MD5 or SHA-1.
    - » Operating system has built-in mechanism to generate random number.

Comparison of PRNG and TRNG

	Pseudorandom Number Generators	True Random Number Generator
Efficiency	Very Efficient	Generally inefficient
Determinism	Deterministic	Nondeterministic
Periodicity	Periodic	Aperiodic

- TRNG is used for key generation and for some applications where small number of random bits are required.

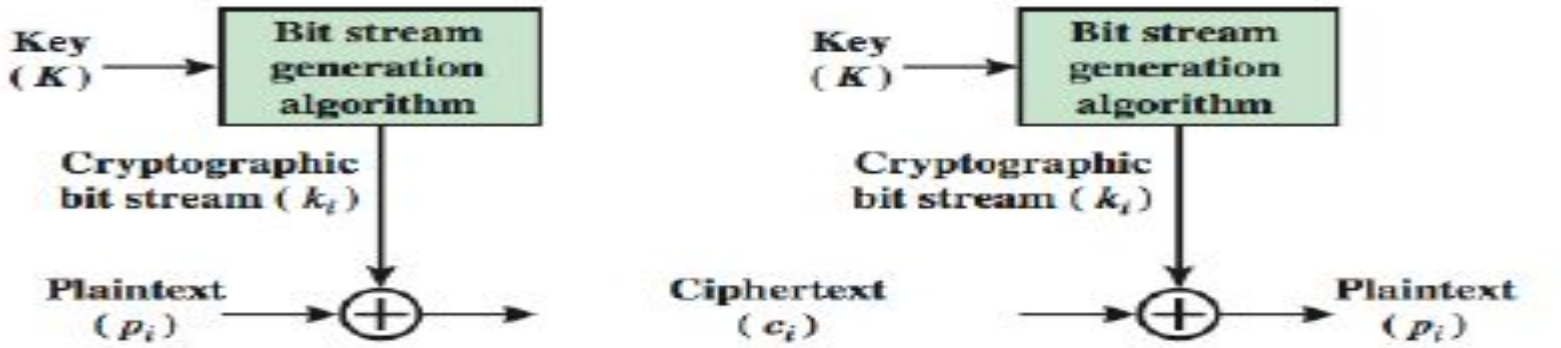
Chapter 8 from your textbook.

CSE4137: Cryptography and Security

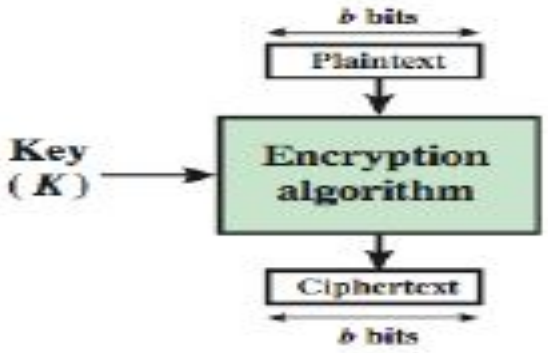
Lecture 7

Block Cipher and The Data Encryption Standard

Block vs Stream Cipher



(a) Stream Cipher Using Algorithmic Bit Stream Generator



(b) Block Cipher

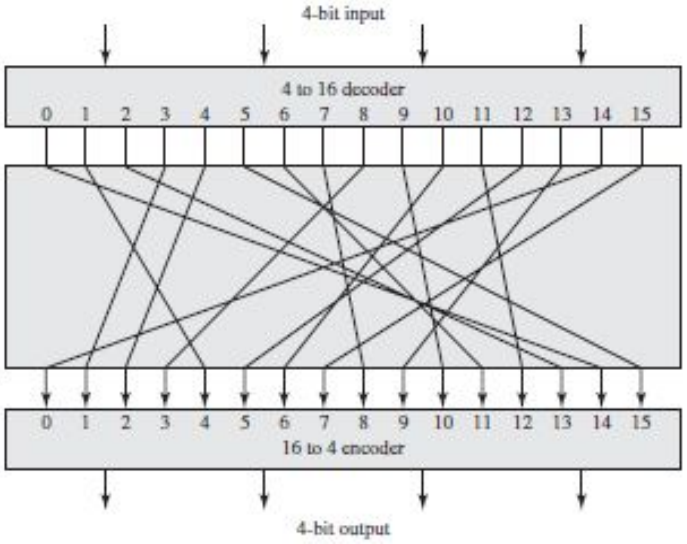
Block vs Stream Cipher

- **Block Cipher:**
  - block ciphers process messages in blocks, each of which is then en/decrypted
  - typically a block size of 64 bits or 128 bits are used.
- **Stream Cipher:**
  - stream ciphers process messages a bit or a byte at a time when en/decrypted

- Many current ciphers are block ciphers
  - better analyzed
  - broader range of applications

Ideal Block Cipher

- In a modern block cipher, a block of ***N*** bits from the plaintext is replaced with a block of ***N*** bits to produce the ciphertext.
- In an ideal block cipher, the relationship between the input blocks and the output block is completely random. But it must be invertible for decryption to work.



Plaintext	Ciphertext
0000	1110
0001	0100
0010	1101
0011	0001
0100	0010
0101	1111
0110	1011
0111	1000
1000	0011
1001	1010
1010	0110
1011	1100
1100	0101
1101	1001
1110	0000
1111	0111

Ciphertext	Plaintext
0000	1110
0001	0011
0010	0100
0011	1000
0100	0001
0101	1100
0110	1010
0111	1111
1000	0111
1001	1101
1010	1001
1011	0110
1100	1011
1101	0010
1110	0000
1111	0101

The Size of the Encryption Key for the Ideal Block Cipher

- The encryption key for the ideal block cipher is the codebook itself, meaning the table that shows the relationship between the input blocks and the output blocks
- With a 64-bit block, we can think of each possible input block as 1 of 2^{64} integers and for each such integer we can specify an output 64-bit block. The size of the codebook will be of size $64 \times 2^{64} \approx 10^{21}$.
- For small block size, ideal block cipher is prone to statistical analysis.

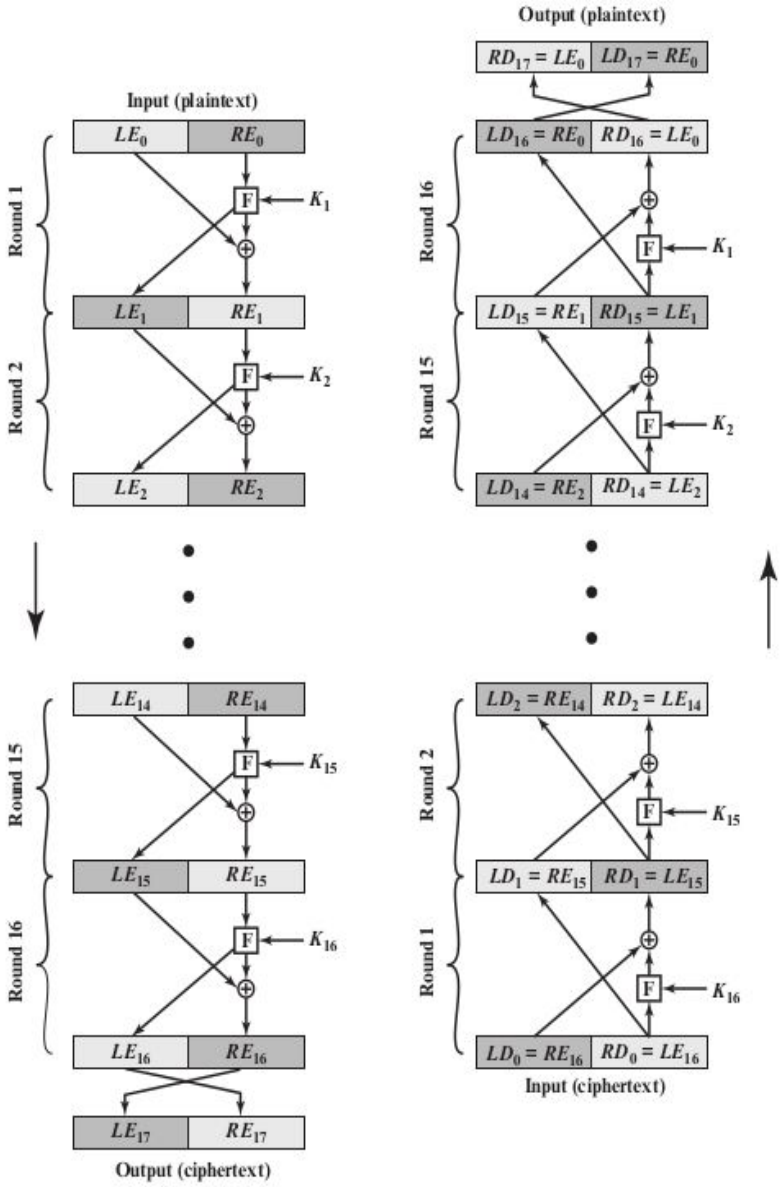
The Feistel Cipher Structure

- Named after the IBM cryptographer Horst Feistel and first implemented in the Lucifer cipher by Horst Feistel and Don Copper smith.
- It is based on Shannon's proposal of 1945, is the structure used by many significant symmetric block ciphers currently in use.
- The idea is to develop a product cipher that alternates ***confusion*** and ***diffusion*** functions.
- A cryptographic system based on Feistel structure uses the same basic algorithm for both encryption and decryption.

Confusion and Diffusion

- Cipher needs to completely obscure statistical properties of original message.
- a one-time pad does this
- more practically Shannon suggested combining *substitution* and *permutation* to obtain:
- **Diffusion** – dissipates statistical structure of plaintext over bulk of ciphertext
- **Confusion** – makes relationship between ciphertext and key as complex as possible.

Feistel Cipher Structure



- ✔ Input to the encryption algorithm
  - ✔ A plaintext block of 2w bits divided into two halves L_0 and R_0.
  - ✔ A Key K
- ✔ N=16 round of processing (although any number can be used).
  - ✔ Round i has input L_{i-1} and R_{i-1} and Key K_i different from K and each other.
- ✔Round function has same structure but is parameterized by K_i.
- ✔Round function performed substitution function on the left half of the data. This is followed by a permutation that interchange the two halves of the data.
- ✔The structure is a particular form of the Substition-Permutation Network (SPN).

Feistel Decryption Algorithm

- ✔ At every round, the intermediate value of the decryption process is equal to the corresponding value of the encryption process with the two halves of the value swapped.
- ✔ Output of the last round of encryption = $RE_{16} \,||\, LE_{16}$ = ciphertext = Input to the first round of decryption process.
  - ✔ Result of encryption of 16th Round:

$$LE_{16} = RE_{15}$$
$$RE_{16} = LE_{15} \oplus \mathrm{F}(RE_{15}, K_{16})$$

  - ✔ On the decryption side:

$$\begin{aligned} LD_1 &= RD_0 = LE_{16} = RE_{15} \\ RD_1 &= LD_0 \oplus \mathrm{F}(RD_0, K_{16}) \\ &= RE_{16} \oplus \mathrm{F}(RE_{15}, K_{16}) \\ &= [LE_{15} \oplus \mathrm{F}(RE_{15}, K_{16})] \oplus \mathrm{F}(RE_{15}, K_{16}) \end{aligned}$$

- ✔ Derivation does not require F to be a reversible function.

Feistel Cipher Design Elements

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

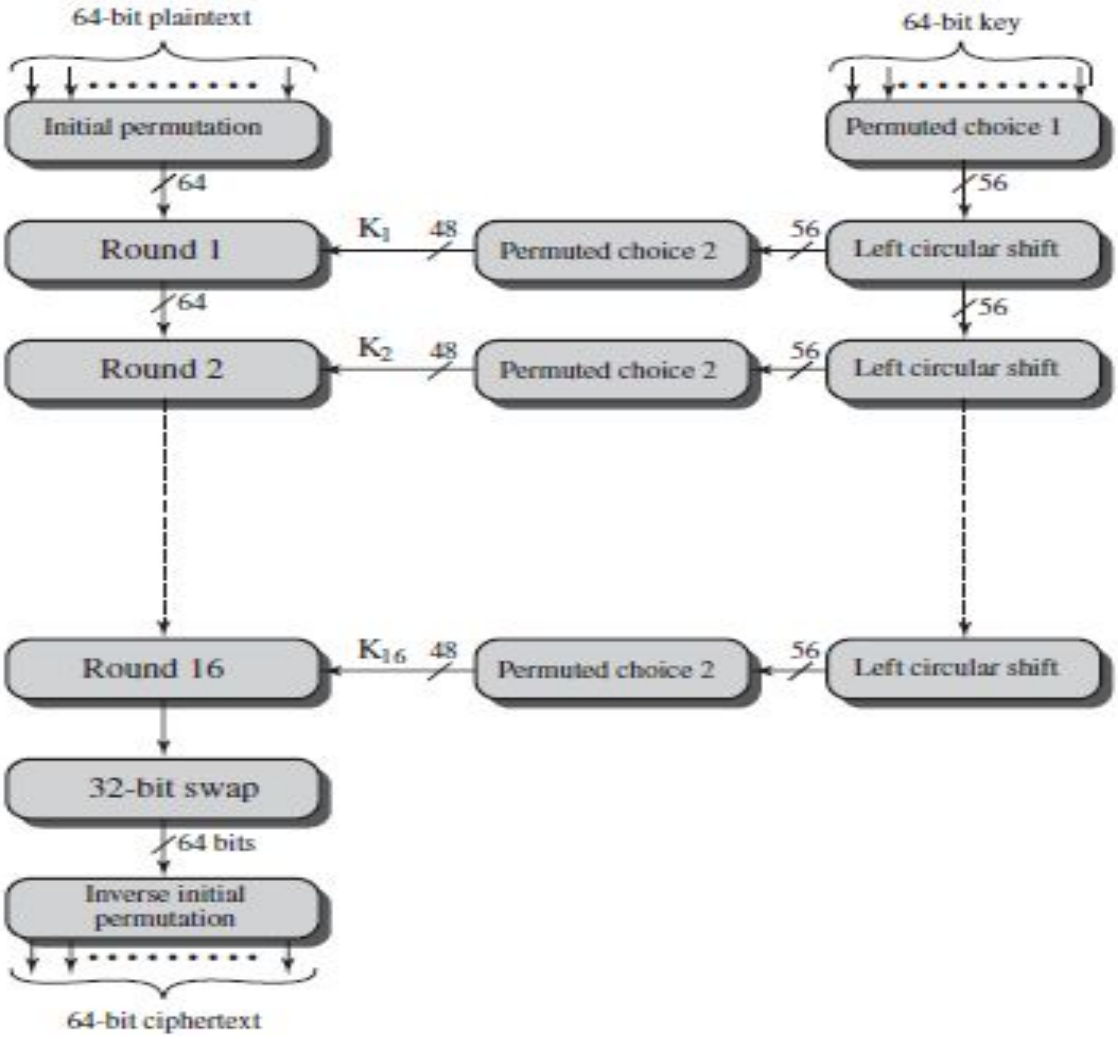
Data Encryption Standard (DES)

- Most widely used block cipher.
- Adopted in 1977 by NBS (now NIST)
  - as FIPS PUB 46
- Encrypts 64-bit data using 56-bit key
- IBM developed Lucifer cipher
  - by team led by Feistel in late 60's
  - used 64-bit data blocks with 128-bit key
- Then redeveloped as a commercial cipher
- In 1973 NBS issued request for proposals for a national cipher standard
- IBM submitted their revised Lucifer which was eventually accepted as the DES.

DES Design Controversy

- Although DES standard is public
- Was considerable controversy over design
  - in choice of 56-bit key (vs Lucifer 128-bit)
  - and because design criteria were classified
- Subsequent events and public analysis show in fact design was appropriate
- Use of DES has flourished
  - especially in financial applications
  - still standardized for legacy application use

DES Encryption Overview



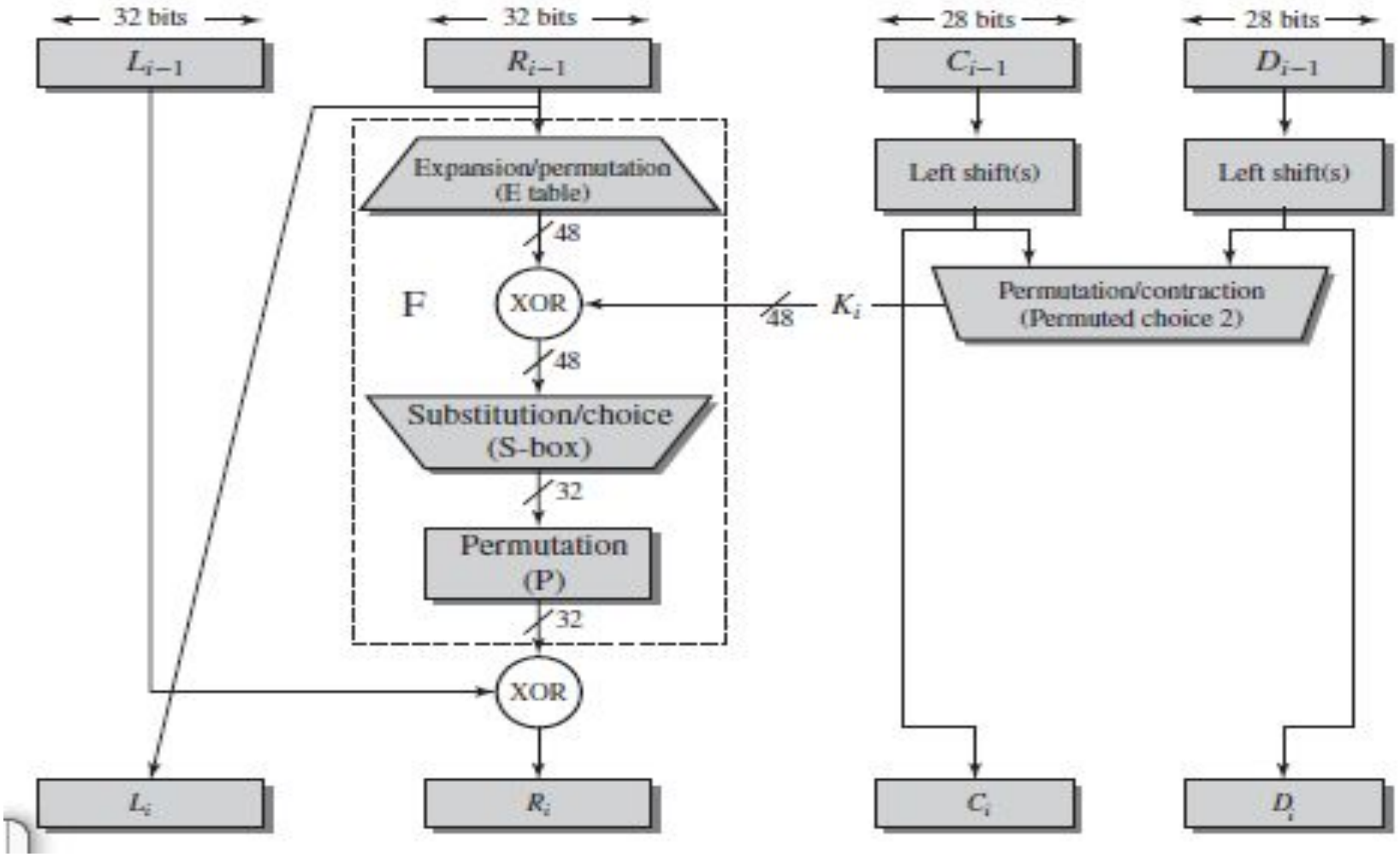
DES Round Structure

- Uses two 32-bit L & R halves
- As for any Feistel cipher can described as:

$$L_i = R_{i-1}$$
$$R_i = L_{i-1} \oplus F(R_{i-1}, K_i)$$

- F takes 32-bit R half and 48-bit subkey:
  - expands R to 48-bits using perm E
  - adds to subkey using XOR
  - passes through 8 S-boxes to get 32-bit result
  - finally permutes using 32-bit perm P

DES Round Structure



Initial Permutation and Inverse Initial Permutation

(a) 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

(b) 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

$M=IP^{-1}(IP(M))=IP^{-1}(X)$

M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8
M_9	M_{10}	M_{11}	M_{12}	M_{13}	M_{14}	M_{15}	M_{16}
M_{17}	M_{18}	M_{19}	M_{20}	M_{21}	M_{22}	M_{23}	M_{24}
M_{25}	M_{26}	M_{27}	M_{28}	M_{29}	M_{30}	M_{31}	M_{32}
M_{33}	M_{34}	M_{35}	M_{36}	M_{37}	M_{38}	M_{39}	M_{40}
M_{41}	M_{42}	M_{43}	M_{44}	M_{45}	M_{46}	M_{47}	M_{48}
M_{49}	M_{50}	M_{51}	M_{52}	M_{53}	M_{54}	M_{55}	M_{56}
M_{57}	M_{58}	M_{59}	M_{60}	M_{61}	M_{62}	M_{63}	M_{64}

$X=IP(M)$

M_{58}	M_{50}	M_{42}	M_{34}	M_{26}	M_{18}	M_{10}	M_2
M_{60}	M_{52}	M_{44}	M_{36}	M_{28}	M_{20}	M_{12}	M_4
M_{62}	M_{54}	M_{46}	M_{38}	M_{30}	M_{22}	M_{14}	M_6
M_{64}	M_{56}	M_{48}	M_{40}	M_{32}	M_{24}	M_{16}	M_8
M_{57}	M_{49}	M_{41}	M_{33}	M_{25}	M_{17}	M_9	M_1
M_{59}	M_{51}	M_{43}	M_{35}	M_{27}	M_{19}	M_{11}	M_3
M_{61}	M_{53}	M_{45}	M_{37}	M_{29}	M_{21}	M_{13}	M_5
M_{63}	M_{55}	M_{47}	M_{39}	M_{31}	M_{23}	M_{15}	M_7

Expansion and Permutation

(c) Expansion Permutation (E)

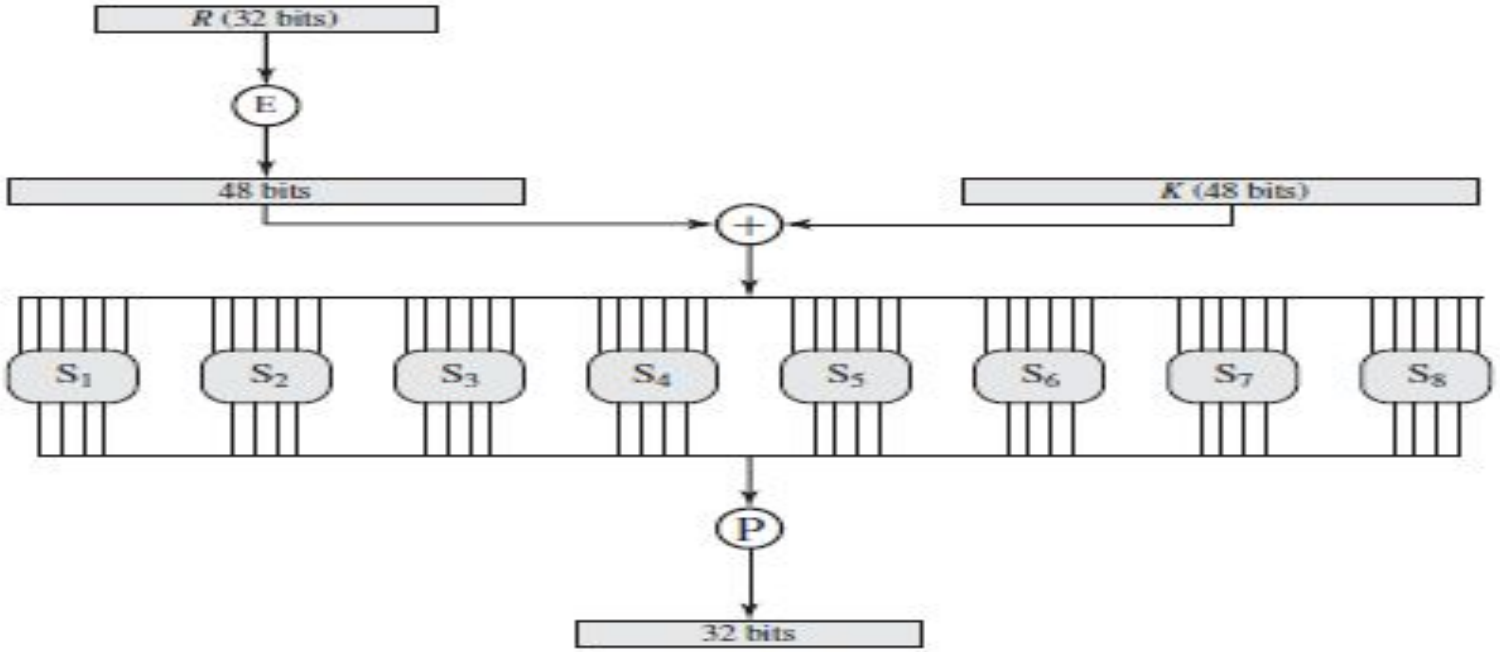
32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

(d) 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

Substitution Boxes S

- Have eight S-boxes which map 6 to 4 bits
  - outer bits 1 & 6 (**row** bits) select 1 row of 4
  - inner bits 2-5 (**col** bits) are substituted
  - result is 8 lots of 4 bits, or 32 bits



Substitution Boxes S

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

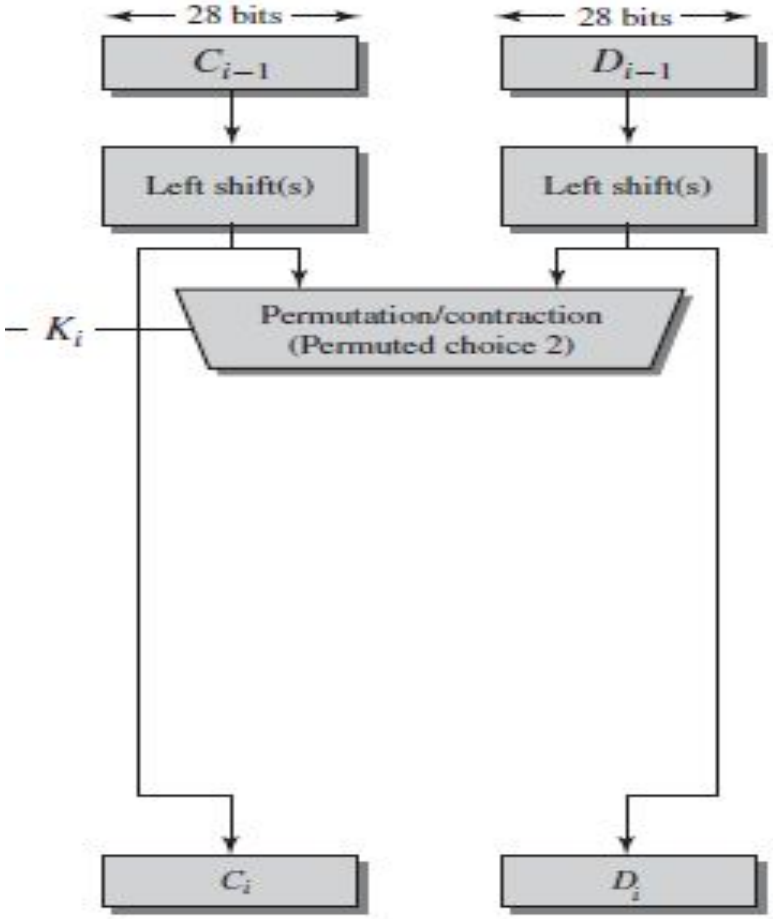
S_7	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12

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

DES Key Schedule

- Forms subkeys used in each round
  - initial permutation of the key (P) which discards every 8^{th} bit to selects 56-bits.
  - Selected 56 bits are divided in two 28-bit halves
- 16 stages consisting of:
  - rotating **each half** separately either 1 or 2 places depending on the **key rotation schedule** K
  - selecting 24-bits from each half & permuting them

DES Key Generation



(a) Input Key

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	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48
49	50	51	52	53	54	55	56
57	58	59	60	61	62	63	64

(b) Permuted Choice One (PC-1)

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

(c) Permuted Choice Two (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

(d) Schedule of Left Shifts

Round Number	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Bits Rotated	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

A DES Example

Table 3.5 DES Example

Round	K_i	L_i	R_i
IP		5a005a00	3cf03c0f
1	1e030f03080d2930	3cf03c0f	bad22845
2	0a31293432242318	bad22845	99e9b723
3	23072318201d0c1d	99e9b723	0bae3b9e
4	05261d3824311a20	0bae3b9e	42415649
5	3325340136002c25	42415649	18b3fa41
6	123a2d0d04262a1c	18b3fa41	9616fe23
7	021f120b1c130611	9616fe23	67117cf2
8	1c10372a2832002b	67117cf2	c11bfc09
9	04292a380c341f03	c11bfc09	887fbc6c
10	2703212607280403	887fbc6c	600f7e8b
11	2826390c31261504	600f7e8b	f596506e
12	12071c241a0a0f08	f596506e	738538b8
13	300935393c0d100b	738538b8	c6a62c4e
14	311e09231321182a	c6a62c4e	56b0bd75
15	283d3e0227072528	56b0bd75	75e8fd8f
16	2921080b13143025	75e8fd8f	25896490
IP^{-1}		da02ce3a	89ecac3b

Plaintext:	02468aceeca86420
Key:	0f1571c947d9e859
Ciphertext:	da02ce3a89ecac3b

Avalanche Effect

- Key desirable property of encryption algorithm.
- Where a change of **one** input or key bit results in changing many bits of the output (ciphertext)
- Small change may provide a way to reduce the size of the plaintext or key space to be searched.
- DES exhibits strong avalanche

Table 3.6 Avalanche Effect in DES: Change in Plaintext

Round		δ
	02468aceeca86420 12468aceeca86420	1
1	3cf03c0fbad22845 3cf03c0fbad32845	1
2	bad2284599e9b723 bad3284539a9b7a3	5
3	99e9b7230bae3b9e 39a9b7a3171cb8b3	18
4	0bae3b9e42415649 171cb8b3ccaca55e	34
5	4241564918b3fa41 ccaca55ed16c3653	37
6	18b3fa419616fe23 d16c3653cf402c68	33
7	9616fe2367117cf2 cf402c682b2cefbc	32
8	67117cf2c11bfc09 2b2cefbc99f91153	33
9	c11bfc09887fbc6c 99f911532eed7d94	32
10	887fbc6c600f7e8b 2eed7d94d0f23094	34
11	600f7e8bf596506e d0f23094455da9c4	37
12	f596506e738538b8 455da9c47f6e3cf3	31
13	738538b8c6a62c4e 7f6e3cf34bc1a8d9	29
14	c6a62c4e56b0bd75 4bc1a8d91e07d409	33
15	56b0bd7575e8fd8f 1e07d4091ce2e6dc	31
16	75e8fd8f25896490 1ce2e6dc365e5f59	32
IP^{-1}	da02ce3a89ecac3b 057cde97d7683f2a	32

Avalanche Effect

Table 3.7 Avalanche Effect in DES: Change in Key

Round		δ
	02468aceeeca86420 02468aceeeca86420	0
1	3cf03c0fbad22845 3cf03c0f9ad628c5	3
2	bad2284599e9b723 9ad628c59939136b	11
3	99e9b7230bae3b9e 9939136b768067b7	25
4	0bae3b9e42415649 768067b75a8807c5	29
5	4241564918b3fa41 5a8807c5488dbe94	26
6	18b3fa419616fe23 488dbe94aba7fe53	26
7	9616fe2367117cf2 aba7fe53177d21e4	27
8	67117cf2c11bfc09 177d21e4548f1de4	32
9	c11bfc09887fbc6c 548f1de471f64dfd	34
10	887fbc6c600f7e8b 71f64dfd4279876c	36
11	600f7e8bf596506e 4279876c399fdc0d	32
12	f596506e738538b8 399fdc0d6d208dbb	28
13	738538b8c6a62c4e 6d208dbbb9bdeeaa	33
14	c6a62c4e56b0bd75 b9bdeeaad2c3a56f	30
15	56b0bd7575e8fd8f d2c3a56f2765c1fb	33
16	75e8fd8f25896490 2765c1fb01263dc4	30
IP^{-1}	da02ce3a89ecac3b ee92b50606b62b0b	30

Strength of DES

- **Key Size:**

Table 5 Average Time Required for Exhaustive Key Search

Key Size (bits)	Cipher	Number of Alternative Keys	Time Required at 10^9 Decryptions/s	Time Required at 10^{13} Decryptions/s
56	DES	$2^{56} \approx 7.2 \times 10^{16}$	2^{55} ns = 1.125 years	1 hour
128	AES	$2^{128} \approx 3.4 \times 10^{38}$	2^{127} ns = 5.3×10^{21} years	5.3×10^{17} years
168	Triple DES	$2^{168} \approx 3.7 \times 10^{50}$	2^{167} ns = 5.8×10^{33} years	5.8×10^{29} years
192	AES	$2^{192} \approx 6.3 \times 10^{57}$	2^{191} ns = 9.8×10^{40} years	9.8×10^{36} years
256	AES	$2^{256} \approx 1.2 \times 10^{77}$	2^{255} ns = 1.8×10^{60} years	1.8×10^{56} years
26 characters (permutation)	Monoalphabetic	$2! = 4 \times 10^{26}$	2×10^{26} ns = 6.3×10^{9} years	6.3×10^{6} years

- **The nature of the DES Algorithm.**
- **Timing Attacks**

Block Cipher Design

- Basic principles still like Feistel's in 1970's
- **Number of rounds:**

  – more is better.

  - General requirement is to make known cryptanalytic effort requires greater effort than a brute-force key attack.

- **Function F:**

  – provides "confusion".

  - should be non-linear (difficult to approximate with a set of linear equations).

  - Strict avalanche criterion (SAC)- an output bit ***j***

- ✔ Chapter 4 of William Stalling book (7^{th} Edition).
- ✔ Chapter 3 of William Stalling book (5^{th} Edition)

Cryptography and Security

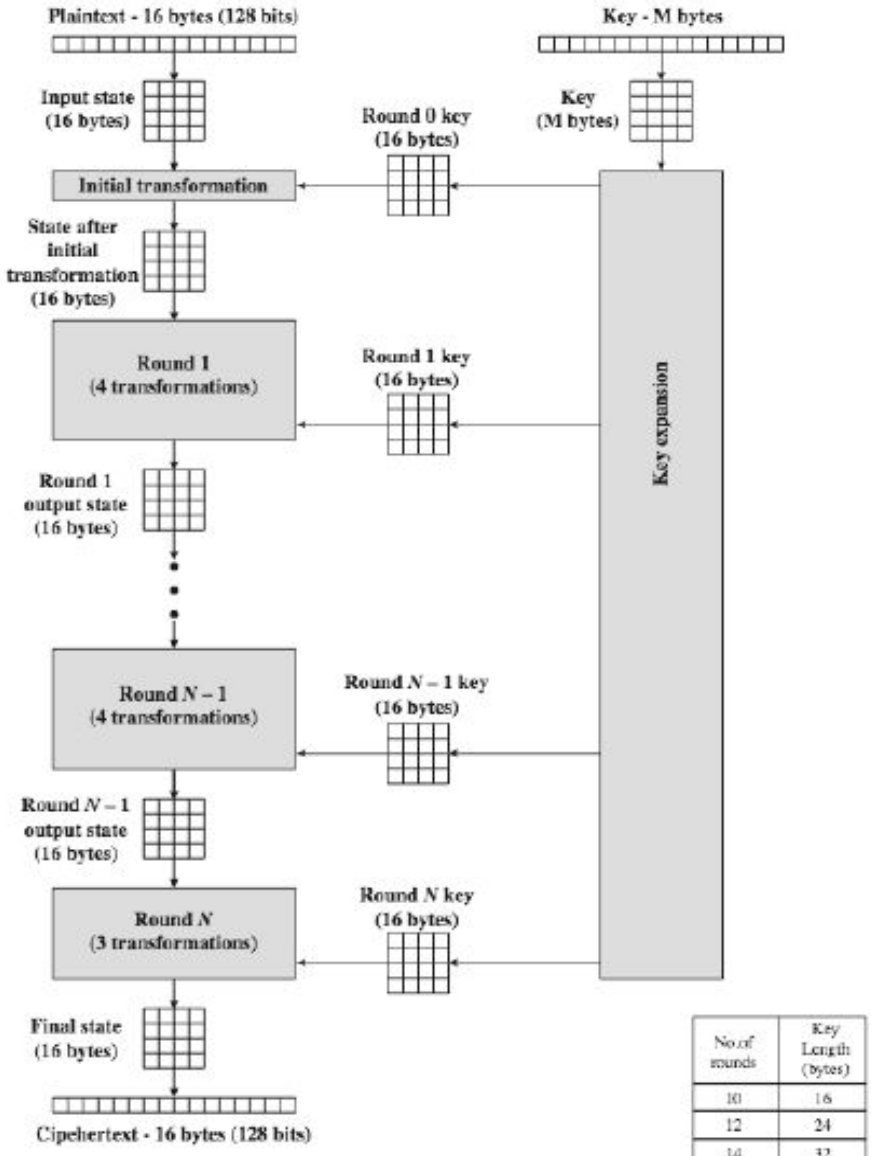
Lecture 8

Advanced Encryption Standard

History of AES

- 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 were 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

AES Structure

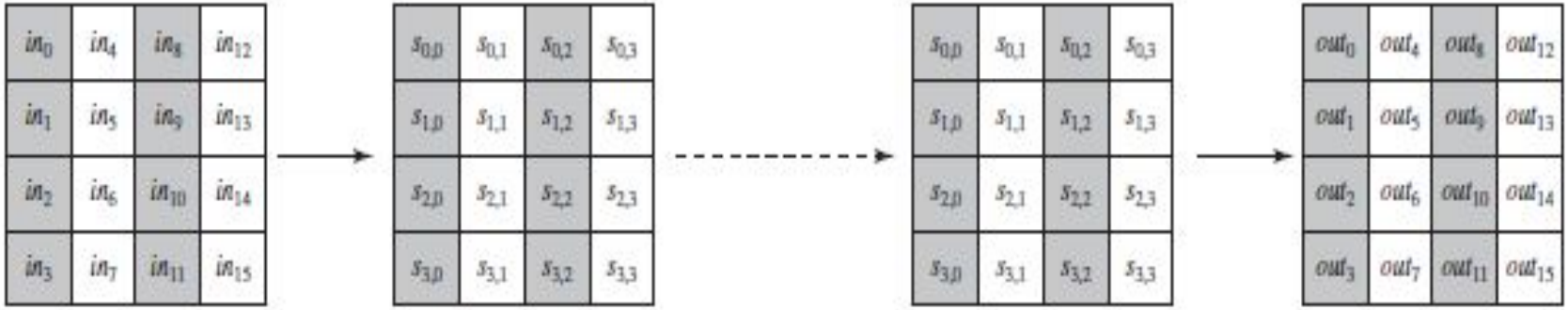


No.of rounds	Key Length (bytes)
10	16
12	24
14	32

- ✔ Plaintext size = 128 bits or 16 bytes
  Key size = 16 (AES-128), 24 (AES-192) and 32 (AES-256) bytes.
- ✔ The cipher consists of N rounds depend on the key length.
- ✔ First N-1 round consists of 4 distinct transformation functions:
  - ✔ SubBytes
  - ✔ ShiftRows
  - ✔ MixColumns
  - ✔ AddRoundKey

  Round N: consists of 3 transformations.
  Round 0: single transformation (AddRoundKey).
- ✔ The key expansion generates N+1 round keys, each of which is a distinct 4×4 matrix.

AES Data Structure



(a) Input, state array, and output

(b) Key and expanded key

Figure 5.2 AES Data Structures

Detailed Structure

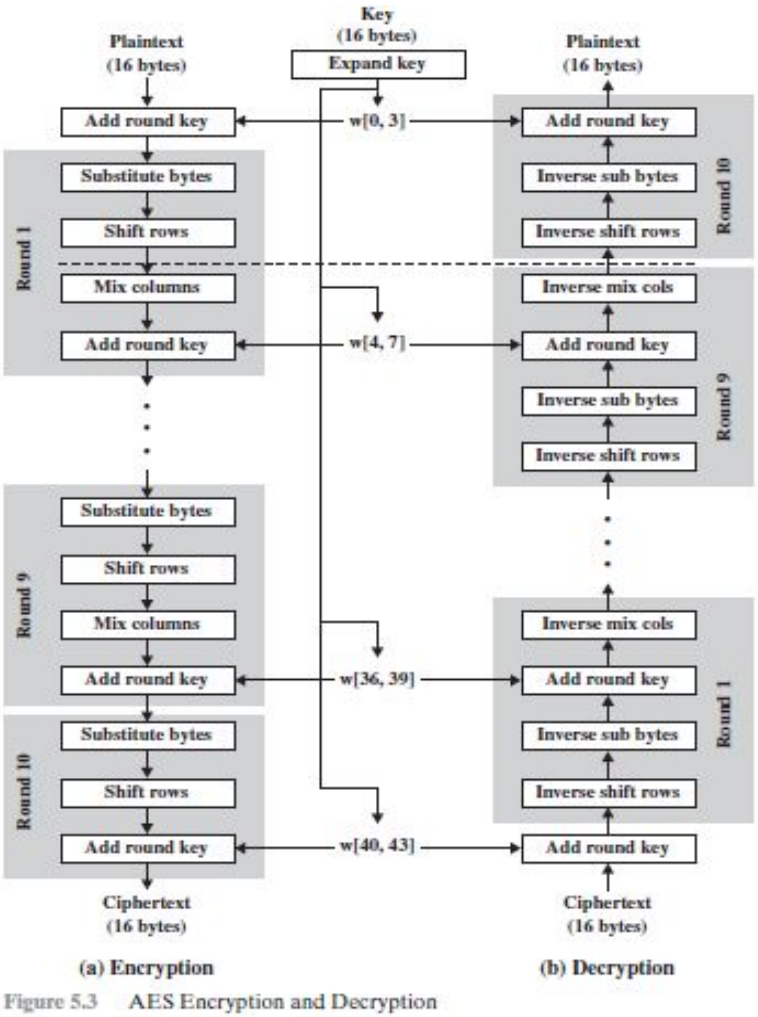


Figure 5.3 AES Encryption and Decryption

1. **Iterative** rather than **Feistel** cipher
2. key expanded into array of forty-four 32-bit words 4 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

Detailed Structure

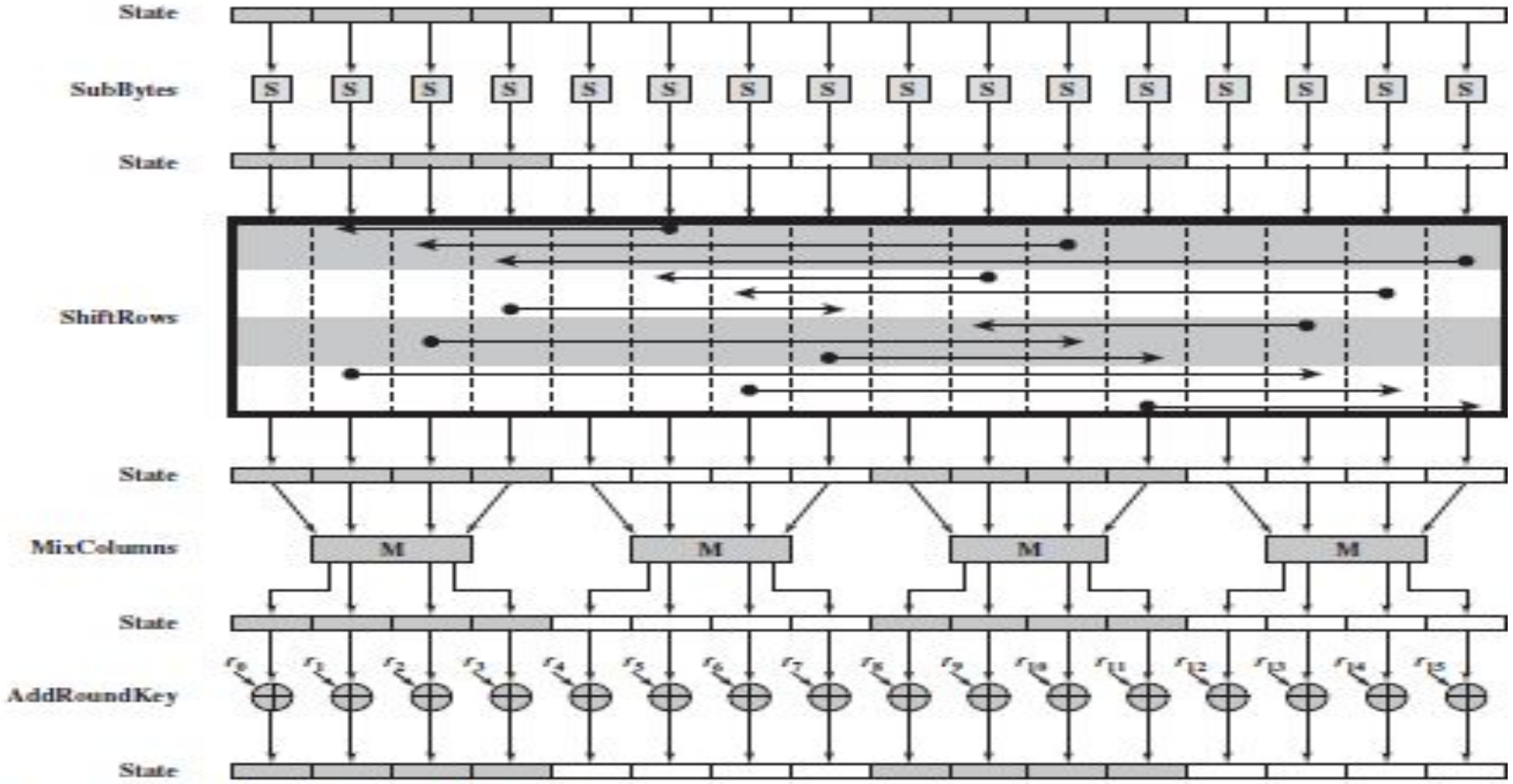
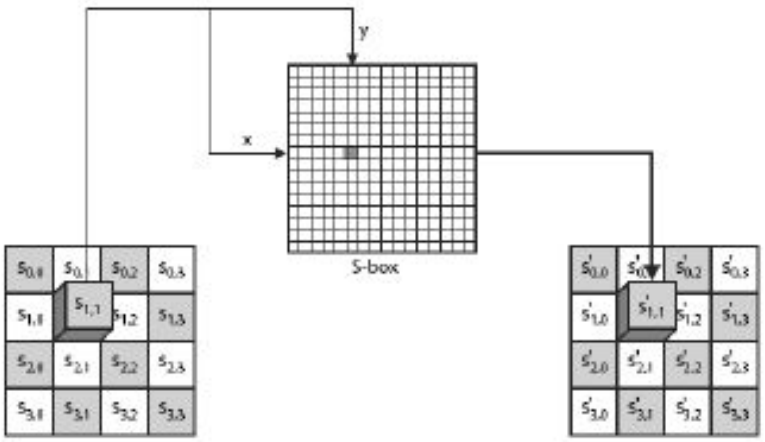


Figure 5.4 AES Encryption Round

Substitute Bytes Transformation

- 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 Transformation



EA	04	65	85	→	87	F2	4D	97
83	45	5D	96		EC	6E	4C	90
5C	33	98	B0		4A	C3	46	E7
F0	2D	AD	C5		8C	D8	95	A6

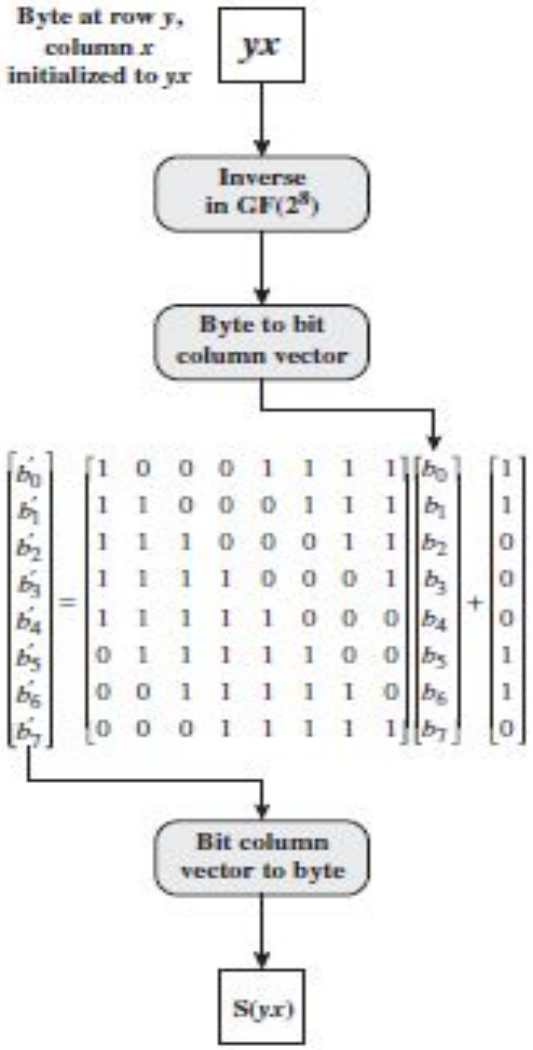
Table 5.2 AES S-Boxes

x \ y	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
5	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
6	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
7	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
8	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
9	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
A	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
B	E7	C8	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
C	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
D	70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1D	9E
E	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
F	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

x \ y	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	52	09	6A	D5	30	36	A5	38	BF	40	A3	9E	81	F3	D7	FB
1	7C	E3	39	82	9B	2F	FF	87	34	8E	43	44	C4	DE	E9	CB
2	54	7B	94	32	A6	C2	23	3D	EE	4C	95	0B	42	FA	C3	4E
3	08	2E	A1	66	28	D9	24	B2	76	5B	A2	49	6D	8B	D1	25
4	72	F8	F6	64	86	68	98	16	D4	A4	5C	CC	5D	65	B6	92
5	6C	70	48	50	FD	ED	B9	DA	5E	15	46	57	A7	8D	9D	84
6	90	D8	AB	00	8C	BC	D3	0A	F7	E4	58	05	B8	B3	45	06
7	D0	2C	1E	8F	CA	3F	0F	02	C1	AF	BD	03	01	13	8A	6B
8	3A	91	11	41	4F	67	DC	EA	97	F2	CF	CE	F0	B4	E6	73
9	96	AC	74	22	E7	AD	35	85	E2	F9	37	E8	1C	75	DF	6E
A	47	F1	1A	71	1D	29	C5	89	6F	B7	62	0E	AA	18	BE	1B
B	FC	56	3E	4B	C6	D2	79	20	9A	DB	C0	FE	78	CD	5A	F4
C	1F	DD	A8	33	88	07	C7	31	B1	12	10	59	27	80	EC	5F
D	60	51	7F	A9	19	B5	4A	0D	2D	E5	7A	9F	93	C9	9C	EF
E	A0	E0	3B	4D	AE	2A	F5	B0	C8	EB	BB	3C	83	53	99	61
F	17	2B	04	7E	BA	77	D6	26	E1	69	14	63	55	21	0C	7D

(b) Inverse S-box

Construction of S-Box



1. Initialize the S-box with the byte values in ascending sequence row by row.

2. Map each byte in the S-box to its multiplicative inverse in GF(2^8). {00} is mapped to itself.

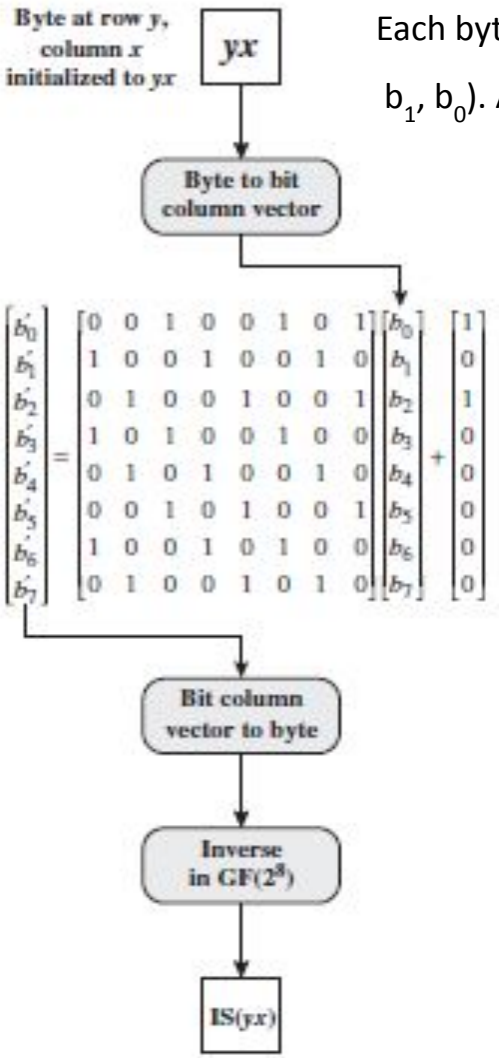
3. Each byte in the S-box consists of 8 bits labeled as (b_7, b_6, b_5, b_4, b_3, b_2, b_1, b_0). Apply the following transformation to each bit

$$b'_i = b_i \oplus b_{(i+4) \bmod 8} \oplus b_{(i+5) \bmod 8} \oplus b_{(i+6) \bmod 8} \oplus b_{(i+7) \bmod 8} \oplus c_i$$

Where, byte c is the value 63, eg. ($c_7 c_6 c_5 c_4 c_3 c_2 c_1 c_0$) = (01100011).

$$\begin{bmatrix} b'_0 \\ b'_1 \\ b'_2 \\ b'_3 \\ b'_4 \\ b'_5 \\ b'_6 \\ b'_7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

Construction of IS-Box



Each byte in the S-box consists of 8 bits labeled as (b_7, b_6, b_5, b_4, b_3, b_2, b_1, b_0). Apply the following transformation to each bit

$$b_i' = b_{(i+2) \bmod 8} \oplus b_{(i+5) \bmod 8} \oplus b_{(i+7) \bmod 8} \oplus d_i$$

Where, d={05}=00000101.

$$\begin{bmatrix} b_0' \\ b_1' \\ b_2' \\ b_3' \\ b_4' \\ b_5' \\ b_6' \\ b_7' \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

**Try example with {95} given in the book by yourself.

InvSubBytes is inverse of SubBytes

- ✔ **X** = matrix of SubBytes and **Y** = matrix of InvSubBytes
- ✔ **C** = matrix for constant **c** and **D** = matrix for constant **d**.
- ✔ From SubBytes, $\mathbf{B' = XB \oplus C}$

It can be proved that $\mathbf{Y(XB \oplus C) \oplus = B=YXB \oplus YC \oplus D}$

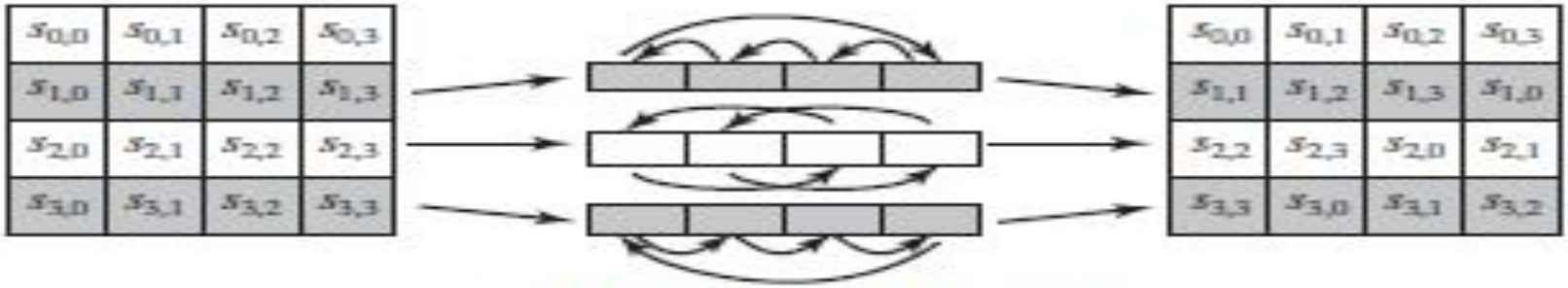
- ✔ **YX** produces identity matrix.
- ✔ ** Try the yourself from the book.

Rationale:

- ✔ Requires low correlation between input bits and output bits and the output is not a linear function of the input.
  - ✔ Non linearity is introduced by the use of multiplicative inverse.
  - ✔ c is chosen so that no S-box(a) = a and S-box(a) = compl(a).
  - ✔ S-box does not self-inverse that is S-box(a) = IS-box(a) is not possible. Eg. S-box(95) = {2A} but IS-box(95) = {AD}.

ShiftRows Transformation

- In forward shift row transformation, circular left shift is performed.
- In inverse shift row transformation, circular right shift is performed.
- This step scatters bytes of a columns to 4 different columns.



(a) Shift row transformation

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

MixColumns Transformation

- Forward mix column transformation operates on each column individually.
- Each byte of a column is mapped into a new value that is a function of all 4 bytes in that column.

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

$$s'_{0,j} = (2 \cdot s_{0,j}) \oplus (3 \cdot s_{1,j}) \oplus s_{2,j} \oplus s_{3,j}$$
$$s'_{1,j} = s_{0,j} \oplus (2 \cdot s_{1,j}) \oplus (3 \cdot s_{2,j}) \oplus s_{3,j}$$
$$s'_{2,j} = s_{0,j} \oplus s_{1,j} \oplus (2 \cdot s_{2,j}) \oplus (3 \cdot s_{3,j})$$
$$s'_{3,j} = (3 \cdot s_{0,j}) \oplus s_{1,j} \oplus s_{2,j} \oplus (2 \cdot s_{3,j})$$

- ✔ Each element in the matrix is the sum of products of elements of one row and one column.
- ✔ The individual additions and multiplications are performed in GF(2^8).

MixColumns 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

$$
\begin{aligned}
(\{02\} \cdot \{87\}) \oplus (\{03\} \cdot \{6E\}) \oplus \{46\} \oplus \{A6\} &= \{47\} \\
\{87\} \oplus (\{02\} \cdot \{6E\}) \oplus (\{03\} \cdot \{46\}) \oplus \{A6\} &= \{37\} \\
\{87\} \oplus \{6E\} \oplus (\{02\} \cdot \{46\}) \oplus (\{03\} \cdot \{A6\}) &= \{94\} \\
(\{03\} \cdot \{87\}) \oplus \{6E\} \oplus \{46\} \oplus (\{02\} \cdot \{A6\}) &= \{ED\}
\end{aligned}
$$

- ✔ In GF(2^8), addition is bitwise XOR operation.
- ✔ Multiplication by 02 is implemented as a 1-bit left shift followed by a conditional XOR with 0001 1011 if the leftmost bit of the original value (prior shift) is 1.

For the first equation, we have $\{02\} \cdot \{87\} = (0000\ 1110) \oplus (0001\ 1011) = (0001\ 0101)$ and $\{03\} \cdot \{6E\} = \{6E\} \oplus (\{02\} \cdot \{6E\}) = (0110\ 1110) \oplus (1101\ 1100) = (1011\ 0010)$. Then,

$$
\begin{aligned}
\{02\} \cdot \{87\} &= 0001\ 0101 \\
\{03\} \cdot \{6E\} &= 1011\ 0010 \\
\{46\} &= 0100\ 0110 \\
\{A6\} &= \underline{1010\ 0110} \\
& 0100\ 0111 = \{47\}
\end{aligned}
$$

Inverse Mix Columns Transformation

$$\begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix} \begin{bmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{bmatrix} = \begin{bmatrix} s'_{0,0} & s'_{0,1} & s'_{0,2} & s'_{0,3} \\ s'_{1,0} & s'_{1,1} & s'_{1,2} & s'_{1,3} \\ s'_{2,0} & s'_{2,1} & s'_{2,2} & s'_{2,3} \\ s'_{3,0} & s'_{3,1} & s'_{3,2} & s'_{3,3} \end{bmatrix}$$

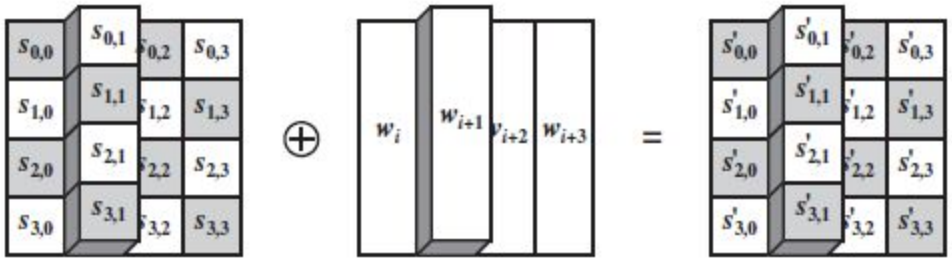
$$\begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix} \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{bmatrix} = \begin{bmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{bmatrix}$$

$$\begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix} \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Try to verify using the example from your book.

AddRoundkey Transformation

- XOR state with 128-bits of the round key
- Processed by column



- Inverse for decryption identical
  - since XOR own inverse, with reversed keys

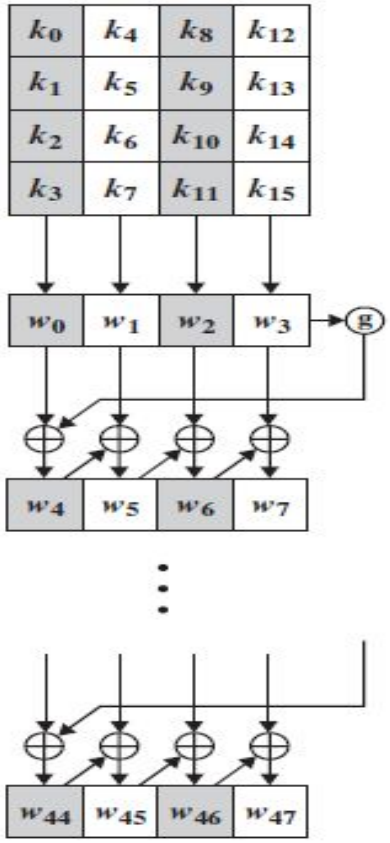
Key Expansion Algorithm

- ✔ Takes as input a four-word (16 bytes) key and produces a linear array of 44 words (176 bytes) which is sufficient to provide a 4 word round key for the initial AddRoundKey stage and each of the 10 rounds of the cipher.

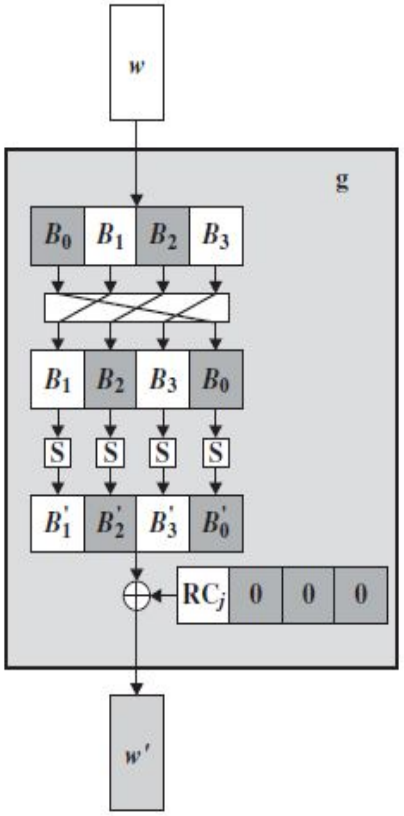
```
KeyExpansion (byte key[16], word w[44])
{
    word temp
    for (i = 0; i < 4; i++)    w[i] = (key[4*i], key[4*i+1],
                                       key[4*i+2],
                                       key[4*i+3]);

    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

Key Expansion Algorithm

- In Rcon[j], the 3 rightmost bytes are always 0.

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

Design Consideration:

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

Do the calculation of generating round key by yourself.

An AES Example

- **Read the example of Key Expansion and Encryption of AES by yourself.**

Avalanche Effect of AES Algorithm

Table 5 Avalanche Effect in AES: Change in Plaintext

Round		Number of Bits that Differ
	0123456789abcdeffedcba9876543210 0023456789abcdeffedcba9876543210	1
0	0e3634aece7225b6f26b174ed92b5588 0f3634aece7225b6f26b174ed92b5588	1
1	657470750fc7ff3fc0e8e8ca4dd02a9c c4a9ad090fc7ff3fc0e8e8ca4dd02a9c	20
2	5c7bb49a6b72349b05a2317ff46d1294 fe2ae569f7ee8bb8c1f5a2bb37ef53d5	58
3	7115262448dc747e5cdac7227da9bd9c ec093dfb7c45343d689017507d485e62	59
4	f867aee8b437a5210c24c1974cffeabc 43efdb697244df808e8d9364ee0ae6f5	61
5	721eb200ba06206dcbd4bce704fa654e 7b28a5d5ed643287e006c099bb375302	68
6	0ad9d85689f9f77bc1c5f71185e5fb14 3bc2d8b6798d8ac4fe36a1d891ac181a	64
7	db18a8ffa16d30d5f88b08d777ba4eaa 9fb8b5452023c70280e5c4bb9e555a4b	67
8	f91b4fbfe934c9bf8f2f85812b084989 20264e1126b219aef7feb3f9b2d6de40	65
9	cca104a13e678500ff59025f3bafaa34 b56a0341b2290ba7dfdfbddcd8578205	61
10	ff0b844a0853bf7c6934ab4364148fb9 612b89398d0600cde116227ce72433f0	58

Avalanche Effect of AES Algorithm

Round		Number of Bits that Differ
	0123456789abcdeffedcba9876543210 0123456789abcdeffedcba9876543210	0
0	0e3634aece7225b6f26b174ed92b5588 0f3634aece7225b6f26b174ed92b5588	1
1	657470750fc7ff3fc0e8e8ca4dd02a9c c5a9ad090ec7ff3fc1e8e8ca4cd02a9c	22
2	5c7bb49a6b72349b05a2317ff46d1294 90905fa9563356d15f3760f3b8259985	58
3	7115262448dc747e5cdac7227da9bd9c 18aeb7aa794b3b66629448d575c7cebf	67
4	f867aee8b437a5210c24c1974cffeabc f81015f993c978a876ae017cb49e7eec	63
5	721eb200ba06206dcbd4bce704fa654e 5955c91b4e769f3cb4a94768e98d5267	81
6	0ad9d85689f9f77bc1c5f71185e5fb14 dc60a24d137662181e45b8d3726b2920	70
7	db18a8ffa16d30d5f88b08d777ba4eaa fe8343b8f88bef66cab7e977d005a03c	74
8	f91b4fbfe934c9bf8f2f85812b084989 da7dad581d1725c5b72fa0f9d9d1366a	67
9	cca104a13e678500ff59025f3bafaa34 0ccb4c66bbfd912f4b511d72996345e0	59
10	ff0b844a0853bf7c6934ab4364148fb9 fc8923ee501a7d207ab670686839996b	53

AES Decryption

- AES decryption is not identical to the encryption.
- Encryption Round:
  - **SubBytes - ShiftRows -MixColumns -AddRoundKey**
- Decryption Round:
  - **InvShiftRows -InvSubBytes -AddRoundKey -InvMixColumns**
  - Interchanging ***InvSubBytes*** and ***InvShiftRows***:

$$\text{InvShiftRows}\,[\text{InvSubBytes}\,(S_i)] = \text{InvSubBytes}\,[\text{InvShiftRows}\,(S_i)]$$

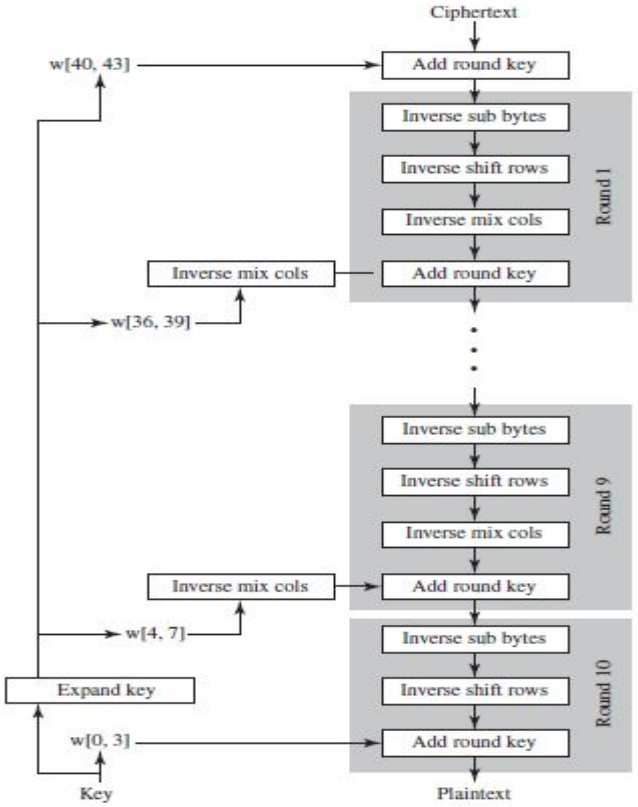
- Interchanging ***AddRoundKeys*** and ***InvMixColumns***:
- For a given state $\boldsymbol{S_i}$ and a round key $\mathbf{w_j}$:

$$\text{InvMixColumns}\;(S_i \oplus w_j) = [\text{InvMixColumns}\,(S_i)] \oplus [\text{InvMixColumns}\,(w_j)]$$

AES Decryption

$$\begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix} \begin{bmatrix} y_0 \oplus k_0 \\ y_1 \oplus k_1 \\ y_2 \oplus k_2 \\ y_3 \oplus k_3 \end{bmatrix} = \begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix} \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \end{bmatrix} \oplus \begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix} \begin{bmatrix} k_0 \\ k_1 \\ k_2 \\ k_3 \end{bmatrix}$$

$$\begin{aligned} &[\{0E\} \cdot (y_0 \oplus k_0)] \oplus [\{0B\} \cdot (y_1 \oplus k_1)] \oplus [\{0D\} \cdot (y_2 \oplus k_2)] \oplus [\{09\} \cdot (y_3 \oplus k_3)] \\ &\quad = [\{0E\} \cdot y_0] \oplus [\{0B\} \cdot y_1] \oplus [\{0D\} \cdot y_2] \oplus [\{09\} \cdot y_3] \oplus \\ &\quad\quad [\{0E\} \cdot k_0] \oplus [\{0B\} \cdot k_1] \oplus [\{0D\} \cdot k_2] \oplus [\{09\} \cdot k_3] \end{aligned}$$



Cryptography and Security

Lecture 9

Block Cipher Operation

Multiple Encryption and DES

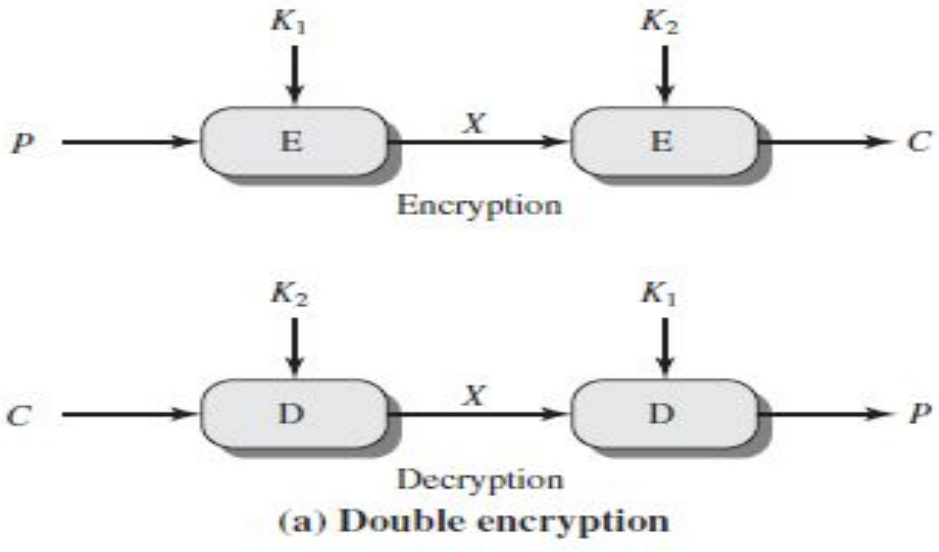
- A replacement for DES was needed
  - demonstrated exhaustive key search attacks
- AES is a new cipher alternative
- Prior to this alternative was to use multiple encryption with DES implementations
- Triple-DES is the chosen form

Double DES

- Given a plaintext ***P***, two encryption key K_1 and K_2, ciphertext ***C*** is generated as

$$C = E(K_2, E(K_1, P))$$

- Decryption requires the keys be applied in reverse order:



(a) Double encryption

Issues with Double DES

- Assumption of $E(K_2, E(K_1, P)) = E(K_3, P)$
  - With 2^{64} possible inputs, the number of mappings that generate a permutation of the input blocks is $(2^{64})! > 10^{1020}$
  - DES defines one mapping for each different key, thus DES uses 2^{56} mappings.
  - If DES is used twice with different keys, it will produce one of the many mappings that are not defined by a single application of DES.

Issues with Double DES

- **Meet-in-the-middle attack**: Based on the assumption

  $C = E(K_2, E(K_1, P))$

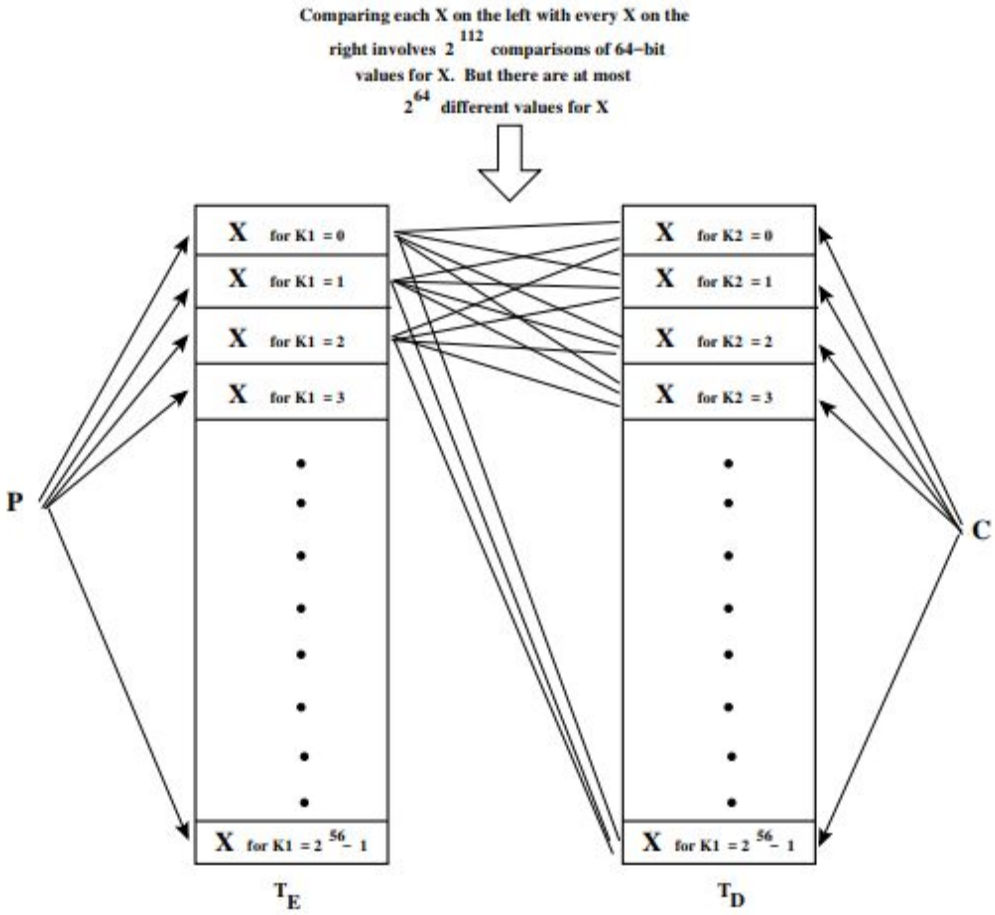
  $X = E(K_1, P) = D(K_2, C)$

- Given a pair ***(P, C)***, the attack proceeds as follows.
  - Encrypt ***P*** for all possible 2^{56} values of K_1 and stores the result in a table.
  - Sort the table by the value of ***X***.
  - Decrypt ***C*** using all possible 2^{56} values of K_2 and for each decryption, check the result against the table for a match.

Issues with Double DES

- If a match occurs, then test (K_1, K_2) for a new plaintext-ciphertext pair.
- If the two keys produce the correct ciphertext, accept them as the correct keys.

Computational Complexity of Meet-in-the Middle Attack (for 2DES)



- 2^{112} comparisons are required to determine which entries are equal. This involves 2^{64} values of X. Thus $2^{112}/2^{64}=2^{48}$ comparisons must involves identical values.
- Therefore, when comparing entries of two tables number of false alarms is 2^{48}.
- For another pair (P', C') we construct $T_{E'}$ and $T_{D'}$ each having 2^{48} entries. So the number of false alarm is $2^{48}/2^{64}=2^{-16}$.
- Thus the probability of getting a single pair (K_1, K_2) that is correct key is $1-2^{-16}$.
- The effort required to make such a comparison is proportional to the size of T_E and T_D, which is 2^{56}, which is comparable to the effort required to break the single DES.

Triple DES with Two Keys

- Defense to the ***meet-in-the-middle*** attack.
- $C = E(K_3, D(K_2, E(K_1, P)))$

  $P = D(K_1, E(K_2, D(K_3, C)))$
- Key length = 168 bits which may be unwieldy for some applications.
- Used in a number of Internet-based applications such as PGP and S/MIME.

Triple DES with Two Keys

- The function follows an encrypt-decrypt-encrypt (EDE) sequence:

$$C = E(K_1, D(K_2, E(K_1, P)))$$

$$P = D(K_1, E(K_2, D(K_1, C)))$$

- Use of decryption allows users of 3DES to decrypt data encrypted by the users of the older single DES:

$$C = E(K_1, D(K_1, E(K_1, P))) = E(K_1, P)$$

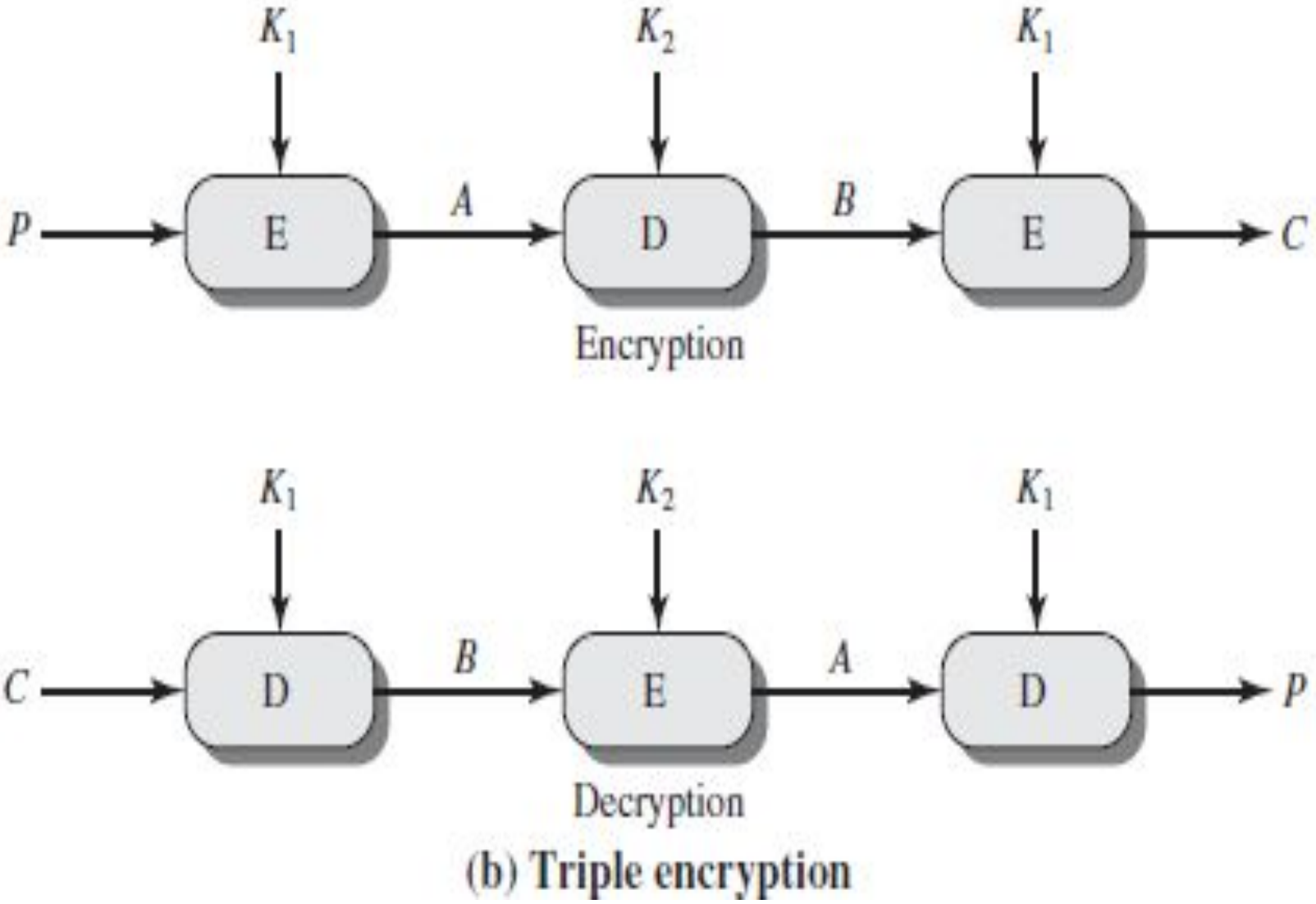
$$P = D(K_1, E(K_1, D(K_1, C))) = D(K_1, C)$$

- 3DES with two keys is a relatively popular alternatives to DES and has been adopted in key management standards ANSI X9.17 and ISO 8732.

Triple DES with Two Keys

- A decryption stage between two encryption stages does not weaken the resulting cryptographic system in any way.
  - decryption in DES works in exactly the same manner as encryption. So if you encrypt data with one key and try to decrypt with a different key, the final output will be still be an encrypted version of the original input.

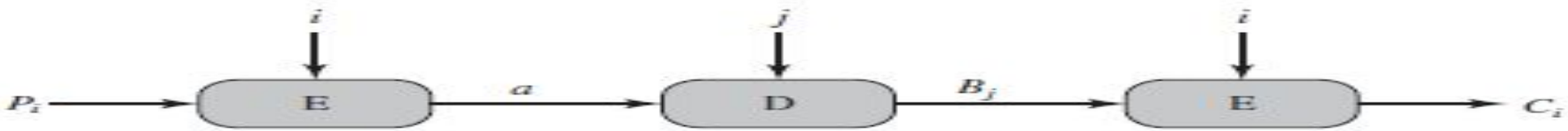
Triple DES with Two Keys



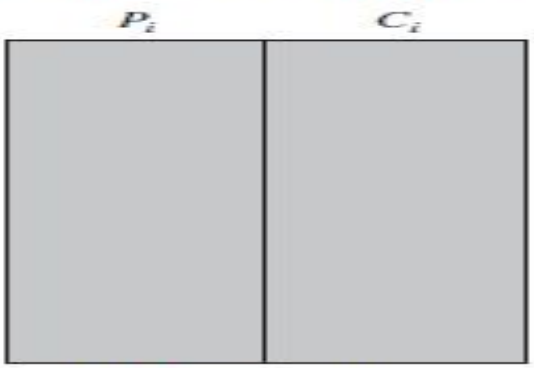
(b) Triple encryption

Attacks on Triple DES with two keys

- It is theoretically possible to extend the meet-in-the-middle attack to 3DES based on two keys.
- If the attacker gets the access of the intermediate value ***A*** for a given plaintext ***P***, breaking the 3DES cipher becomes the same as breaking 2DES with the meet-in-the-middle attack.



(a) Two-key triple encryption with candidate pair of keys



(b) Table of n known plaintext-ciphertext pairs, sorted on P

(c) Table of intermediate values and candidate keys

Figure 6.2 Known-Plaintext Attack on Triple DES

Attacks on Triple DES with two keys

- **Step 1:** The attacker procures ***n*** pairs of ***(P, C)***. These are arranged in a two-column table, with all the ***P***'s in one column and their corresponding ***C***'s in the other column.
- **Step 2:** The attacker now chooses an arbitrary value ***a*** for ***A***. The attacker figures out the plaintext that will result in ***a*** for every possible key $K_1 = i: P_i = D(i, a)$

  If a P_i matched in Table I, creates an entry in Table II consisting of value of K_1 and ***B*** such that $B = D(i, C)$ for ***C*** that corresponds to P_i.

  Sort Table II by ***B***.
- **Step 3:** For K_1, search for each 2^{56} possible K_2 using $B_j = D(j, a)$. Search B_j in Table II and if there is a match then (i, j) is the candidate value for (K_1, K_2).
- **Step 4:** Test for other few plaintext-ciphertext pairs. If (K_1, K_2) produces the desired ciphertext, the task is complete.

Attacks on Triple DES with two keys

- For a given ***(P, C)***, the probability of selecting unique ***a*** that leads to success is $1/2^{64}$.
- Thus given ***n (P, C)*** pairs, the probability of success for a single selected value of ***a*** is $n/2^{64}$.
- The expected number of draws required to draw one red ball from a bin containing ***n*** red balls and ***N − n*** green balls is ***(N + 1)/(n + 1)*** if the balls are NOT replaced.
- So given the ***n*** pairs for ***(P, C)***, the number of different possible values for ***a*** that we may have to try is

$$\frac{2^{64}+1}{n+1} \approx \frac{2^{64}}{n}$$

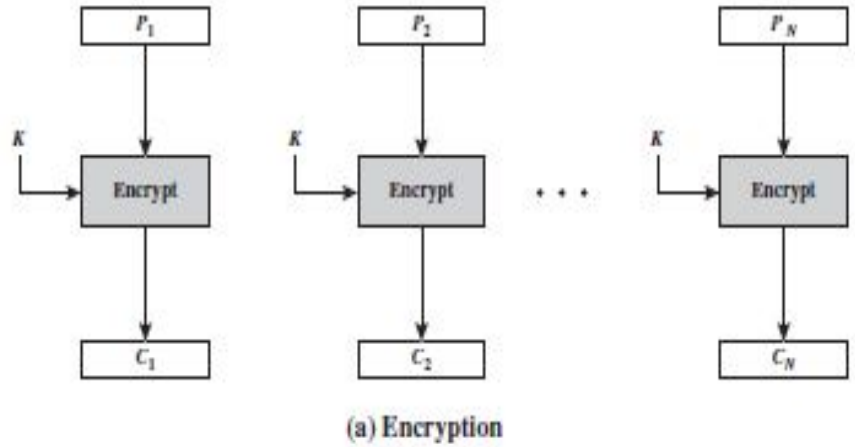
- The expected running time of the attack is on the order of

$$(2^{56})\frac{2^{64}}{n} = 2^{120-\log_2 n}$$

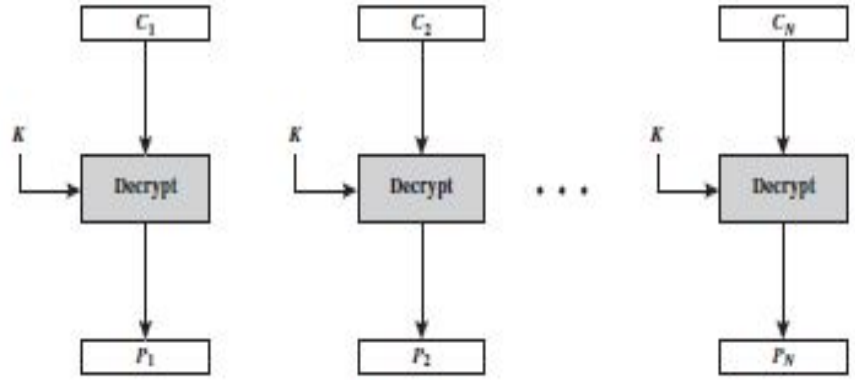
Block Cipher Modes of Operation

- A block cipher takes a fixed-length block of text of length ***b*** and a key as input and produces a ***b***-bit block of ciphertext.
- If the amount of plaintext is greater than ***b*** bits, the plaintext is divided into blocks of ***b*** bits.
- When multiple blocks of plaintext is encrypted with same key, a number of security issues arises.
- To apply a block cipher in a variety of applications, five modes of operation have been defined:
  - Electronic Code Book
  - Cipher Block Chaining
  - Cipher Feedback
  - Output Feedback
  - Counter (CTR)

Electronic Code Book



(a) Encryption



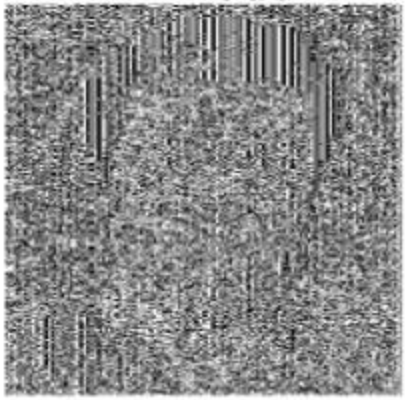
(b) Decryption

- Ideal for short amount of data, such as to transmit AES and DES key.
- If same b-bit blocks of plaintext appears more than once in the message, it always produce the same ciphertext.
- For lengthy messages, if the message is highly structured, it may be possible to exploit these regularities.

Electronic Code Book

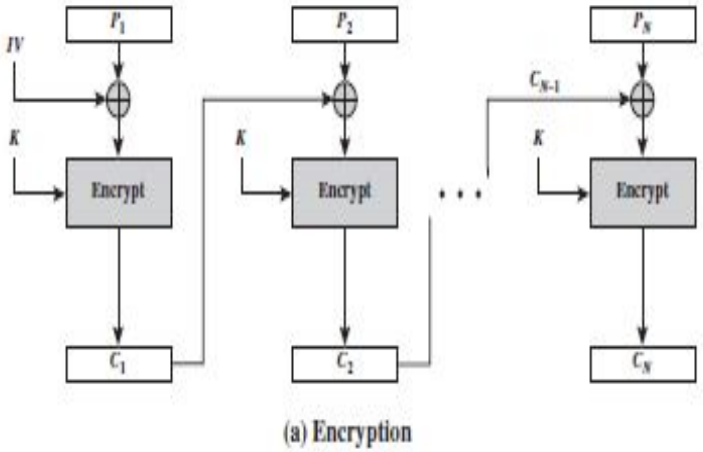
An example plaintext

Encrypted with AES in ECB mode

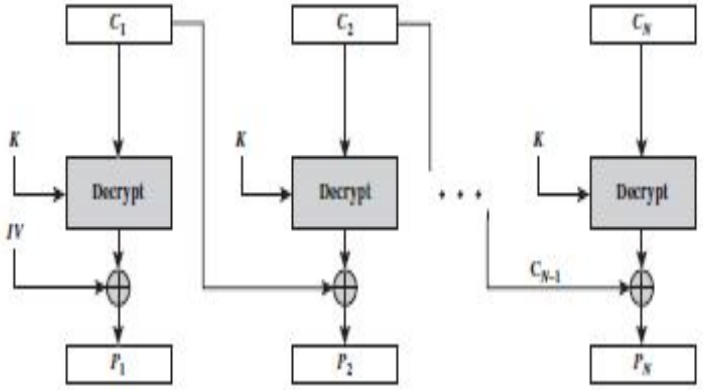


Courtesy: B. Preneel

Cipher Block Chaining Mode



(a) Encryption



(b) Decryption

Figure 6.4 Cipher Block Chaining (CFB) Mode

- Overcome the security deficiencies of ECB.

$$C_j = \mathrm{E}(K, [C_{j-1} \oplus P_j])$$

$$\mathrm{D}(K, C_j) = \mathrm{D}(K, E(K, [C_{j-1} \oplus P_j]))$$
$$\mathrm{D}(K, C_j) = C_{j-1} \oplus P_j$$
$$C_{j-1} \oplus \mathrm{D}(K, C_j) = C_{j-1} \oplus C_{j-1} \oplus P_j = P_j$$

- IV must be protected against unauthorized changes. This could be done by sending IV using ECB encryption.

Cipher Block Chaining Mode

- If an opponent is able to fool the receiver into using a different value for IV, the opponent is able to invert selected bits in the first block of plaintext.

$$C_1 = E(K, [IV \oplus P_1])$$
$$P_1 = IV \oplus D(K, C_1)$$

$$P_1[i] = IV[i] \oplus D(K, C_1)[i]$$

$$P_1[i]' = IV[i]' \oplus D(K, C_1)[i]$$

Cipher Block Chaining Mode

- IV can be generated in two ways:
  - Nonce: unique value unique for each execution of encryption algorithm. May be a counter, a timestamp or a message number.
  - Generate a random block using a random number generator.
- Applications:
  - Suitable for encrypting messages of length greater than b bits.
  - Used for achieving confidentiality and authentication.

Cipher Feedback Mode

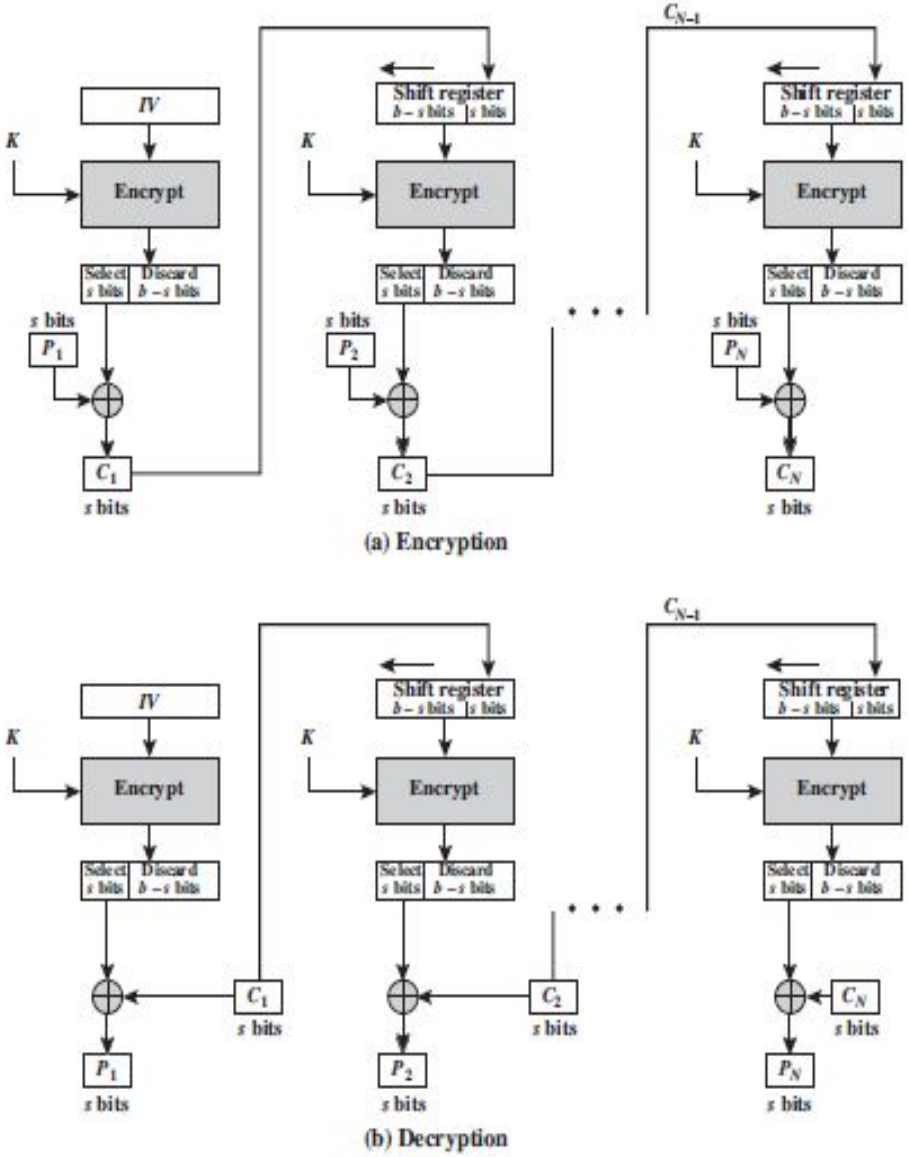


Figure 6.5 s-bit Cipher Feedback (CFB) Mode

- message is treated as a stream of bits.
  - Eliminates the need to pad a message to be an integral number of blocks.
  - If a character stream is being transmitted, each character can be encrypted and transmitted immediately using a character-oriented stream cipher.

Cipher Feedback Mode

- Encryption function is used in decryption.

$$C_1 = P_1 \oplus \text{MSB}_s[\text{E}(K, \text{IV})]$$

$$P_1 = C_1 \oplus \text{MSB}_s[\text{E}(K, \text{IV})]$$

- In encryption, each forward cipher function depends on the result of the previous forward cipher function. Thus multiple forward cipher operations cannot be performed in parallel.
- In decryption, the forward cipher functions can be performed in parallel if the input blocks are first constructed from the IV and ciphertext.

Output Feedback Mode

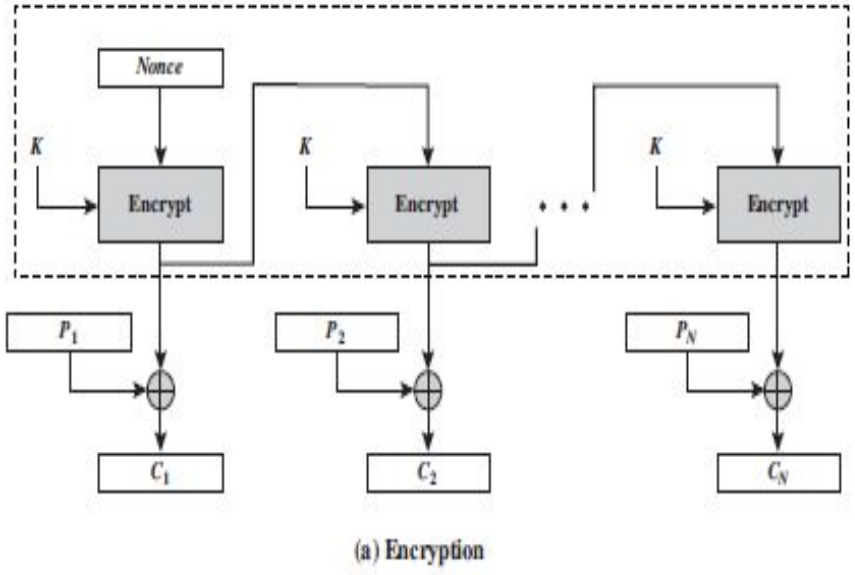


Figure 6.6 Output Feedback (OFB) Mode

- $C_j = P_j \text{ XOR } E(K, O_{j-1})$

  $O_{j-1} = E(K, O_{j-2})$
- $C_j = P_j \text{ XOR } E(K, C_{j-1} \text{ XOR } P_{j-1})$

  $P_j = C_j \text{ XOR } E(K, C_{j-1} \text{ XOR } P_{j-1})$
- If the size of the last block is $u<b$ bits, the most significant u bits of the last output block O_N are used for XOR operation. Remaining bits of the last output block are discarded.
- In OFB, the IV must be a nonce which is unique for each execution of the encryption operation. Otherwise, identical plaintext at identical position of two different message will produce same ciphertext.

Output Feedback Mode

- In OFB, bit errors in transmission do not propagate. If bit error occurs in C_1, the recovered value of P_1 is affected.
- Vulnerable to message stream modification attack. Changing one bit in the ciphertext directly affect the corresponding bit in the plaintext. It is possible for opponent to make changes to some portion of data and then make changes to checksum. The receiver cannot detect the changes in the plaintext.

Counter Mode

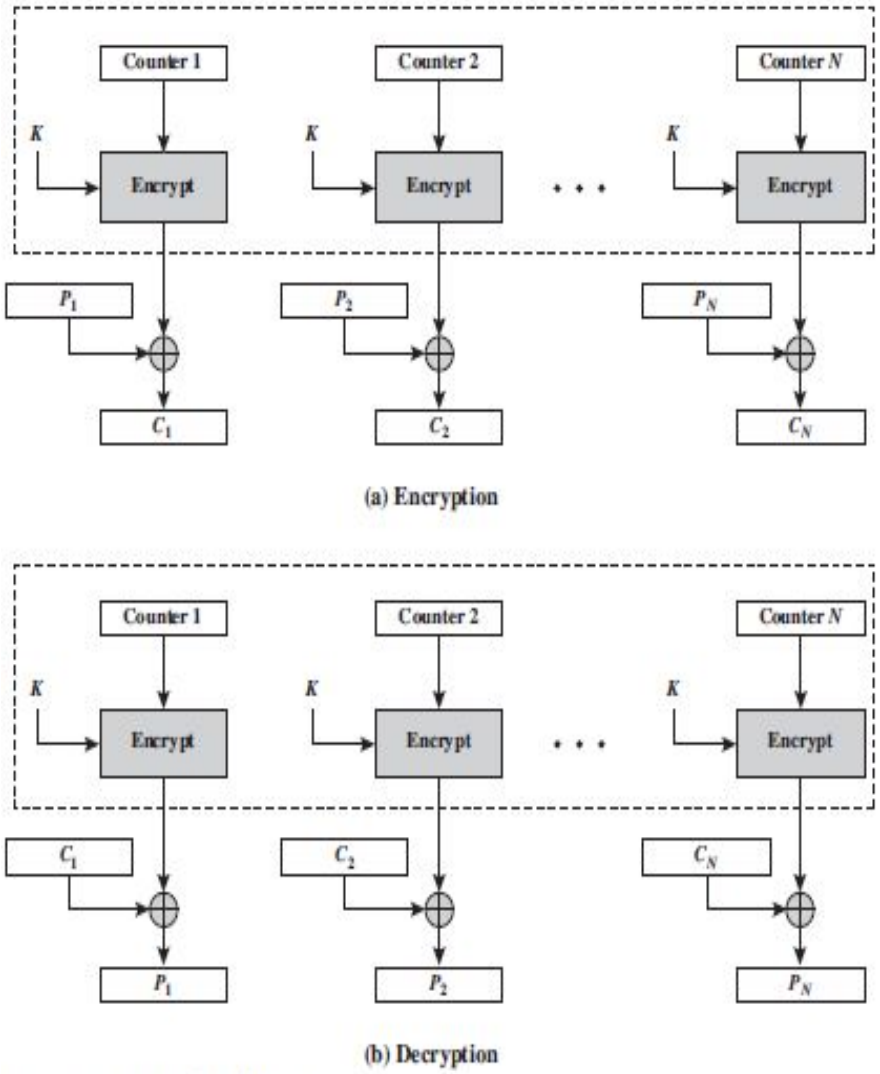


Figure 6.7 Counter (CTR) Mode

CTR	$C_j = P_j \oplus E(K, T_j) \quad j = 1, \ldots, N-1$ $C_N^* = P_N^* \oplus \text{MSB}_u[E(K, T_N)]$	$P_j = C_j \oplus E(K, T_j) \quad j = 1, \ldots, N-1$ $P_N^* = C_N^* \oplus \text{MSB}_u[E(K, T_N)]$

- The counter value must be a nonce. That is, all T_i across all messages must be unique.
- If a plaintext block is encrypted using a given counter value is known, then the output of the encryption block is known from the associated ciphertext. This output allows other plaintext blocks that are encrypted using the same counter value to be easily recovered from the associated ciphertext blocks.

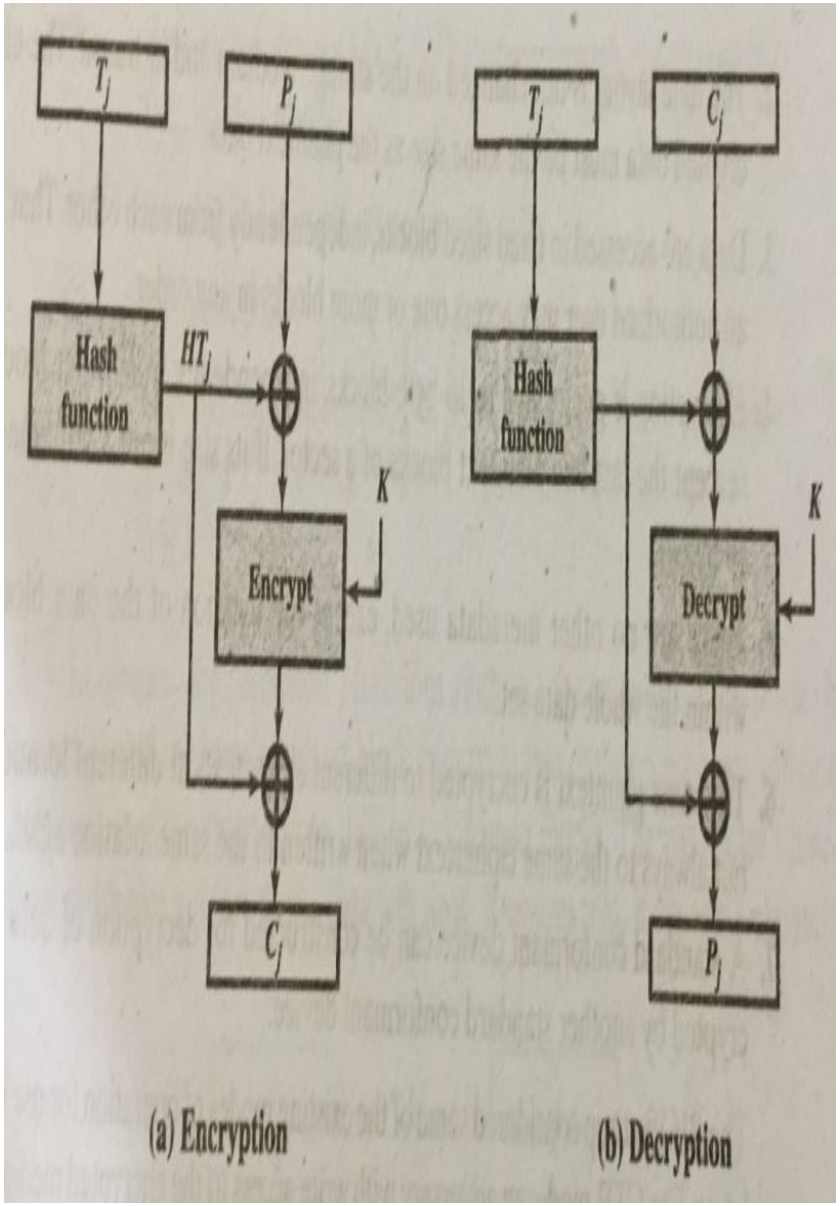
Counter Mode

- Encryption and decryption can be performed in parallel. The throughput is only limited by the amount of parallelism that is achieved.
- For the scope of parallel execution, processor can support parallel features such as aggressive pipelining, multiple instruction dispatch per clock cycle, a large number of registers and SIMD instruction, etc can be effectively utilized.
- Preprocessing of the encryption algorithm are possible.
- Random access is possible as chaining mode is not used.
- Unlike ECB and CBC modes, CTR mode requires only the implementation of the encryption algorithm.

XTS-AES Mode

- In 2010, NIST approved additional block cipher mode of operation, XTS-AES.
- This mode is also an IEEE standard: IEEE Std 1619-2007.
- A method for encryption for data stored in sector-based devices.

Tweakable Block Cipher



- Used as a basis of XTS-AES mode.
- The purpose of tweak is to introduce variability and it is not secret.
- ***C= H(T) XOR E(K, H(T) XOR P)***
- For decryption,

  H(T) XOR C = E(K, H(T) XOR P)

  D[K, H(T) XOR C] = H(T) XOR P

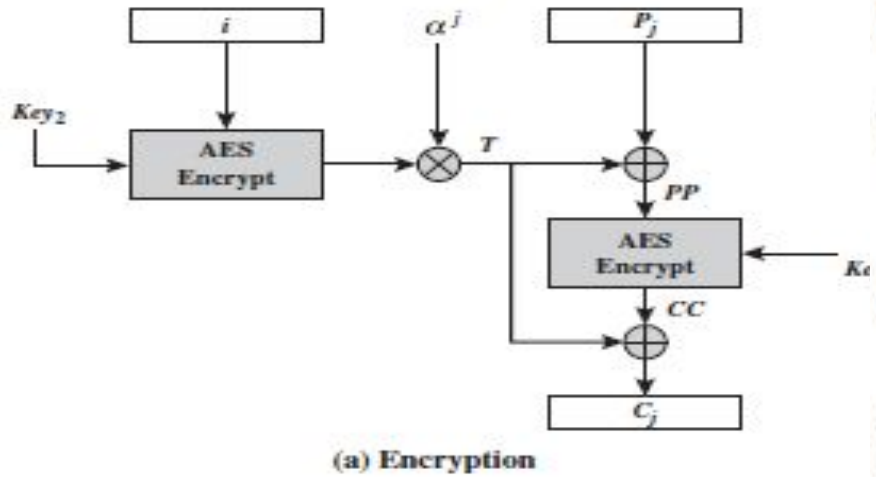
  H(T) XOR D[K, H(T) XOR C]=

  H(T) XOR H(T) XOR P = P.
- Tweak cipher uses ECB by removing the security issue of ECB.

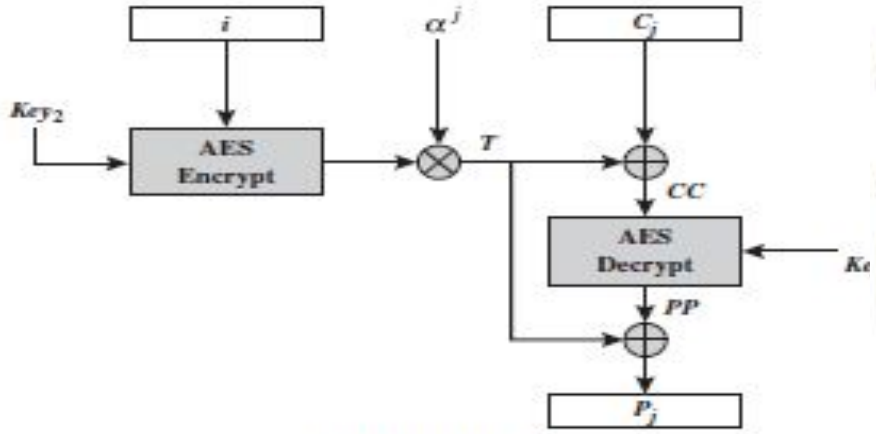
Storage Encryption Requirements

1. The ciphertext is freely available for an attacker. Among the circumstances that lead to this situation:
   a. A group of users has authorized access to a database. Some of the records in the database are encrypted so that only specific users can successfully read/write them. Other users can retrieve an encrypted record but are unable to read it without the key.
   b. An unauthorized user manages to gain access to encrypted records.
   c. A data disk or laptop is stolen, giving the adversary access to the encrypted data.
2. The data layout is not changed on the storage medium and in transit. The encrypted data must be the same size as the plaintext data.
3. Data are accessed in fixed sized blocks, independently from each other. That is, an authorized user may access one or more blocks in any order.
4. Encryption is performed in 16-byte blocks, independently from other blocks (except the last two plaintext blocks of a sector, if its size is not a multiple of 16 bytes).
5. There are no other metadata used, except the location of the data blocks within the whole data set.
6. The same plaintext is encrypted to different ciphertexts at different locations, but always to the same ciphertext when written to the same location again.
7. A standard conformant device can be constructed for decryption of data encrypted by another standard conformant device.

Operation on a Single Block



(a) Encryption

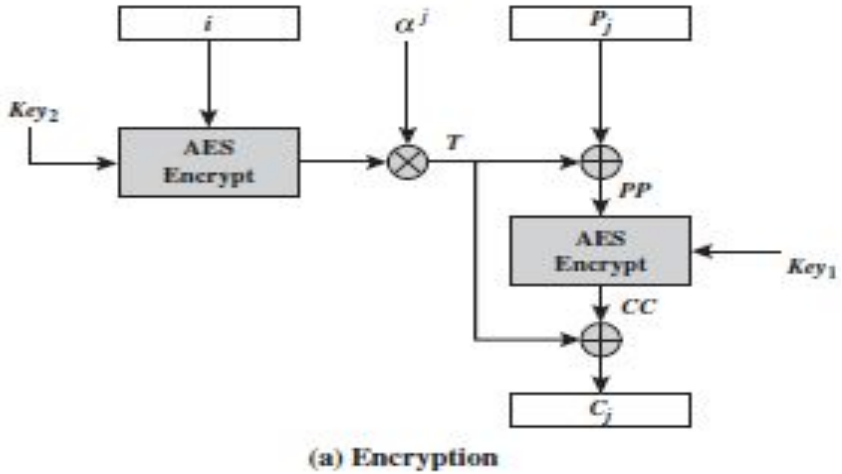


(b) Decryption

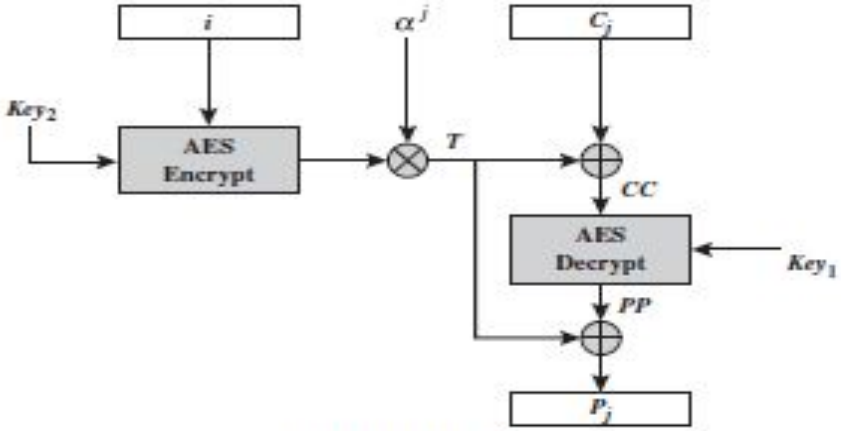
Figure 6.9 XTS-AES Operation on Single Block

Key	The 256 or 512 bit XTS-AES key; this is parsed as a concatenation of two fields of equal size called Key_1 and Key_2, such that $Key = Key_1 \| Key_2$.
P_j	The jth block of plaintext. All blocks except possibly the final block have a length of 128 bits. A plaintext data unit, typically a disk sector, consists of a sequence of plaintext blocks $P_1, P_2, \ldots, P_m$.
C_j	The jth block of ciphertext. All blocks except possibly the final block have a length of 128 bits.
j	The sequential number of the 128-bit block inside the data unit.
i	The value of the 128-bit tweak. Each data unit (sector) is assigned a tweak value that is a nonnegative integer. The tweak values are assigned consecutively, starting from an arbitrary nonnegative integer.
α	A primitive element of $GF(2^{128})$ that corresponds to polynomial x (i.e., $0000\ldots010_2$).
α^j	α multiplied by itself j times, in $GF(2^{128})$.
$\oplus$	Bitwise XOR.
$\otimes$	Modular multiplication of two polynomials with binary coefficients modulo $x^{128} + x^7 + x^2 + x + 1$. Thus, this is multiplication in $GF(2^{128})$.

Operation on a Single Block



(a) Encryption



(b) Decryption

Figure 6.9 XTS-AES Operation on Single Block

XTS-AES block operation	$T = E(K_2, i) \otimes \alpha^j$ $PP = P \oplus T$ $CC = E(K_1, PP)$ $C = CC \oplus T$	$T = E(K_2, i) \otimes \alpha^j$ $CC = C \oplus T$ $PP = D(K_1, CC)$ $P = PP \oplus T$

- For encryption,

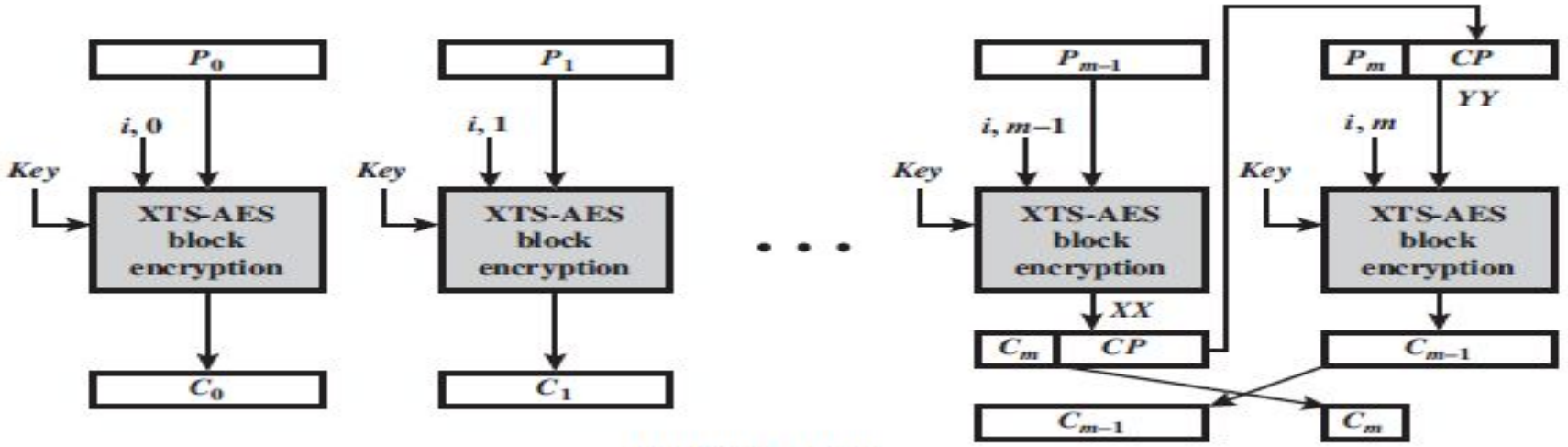
$$C = CC \oplus T = E(K_1, PP) \oplus T = E(K_1, P \oplus T) \oplus T$$

- For decryption,

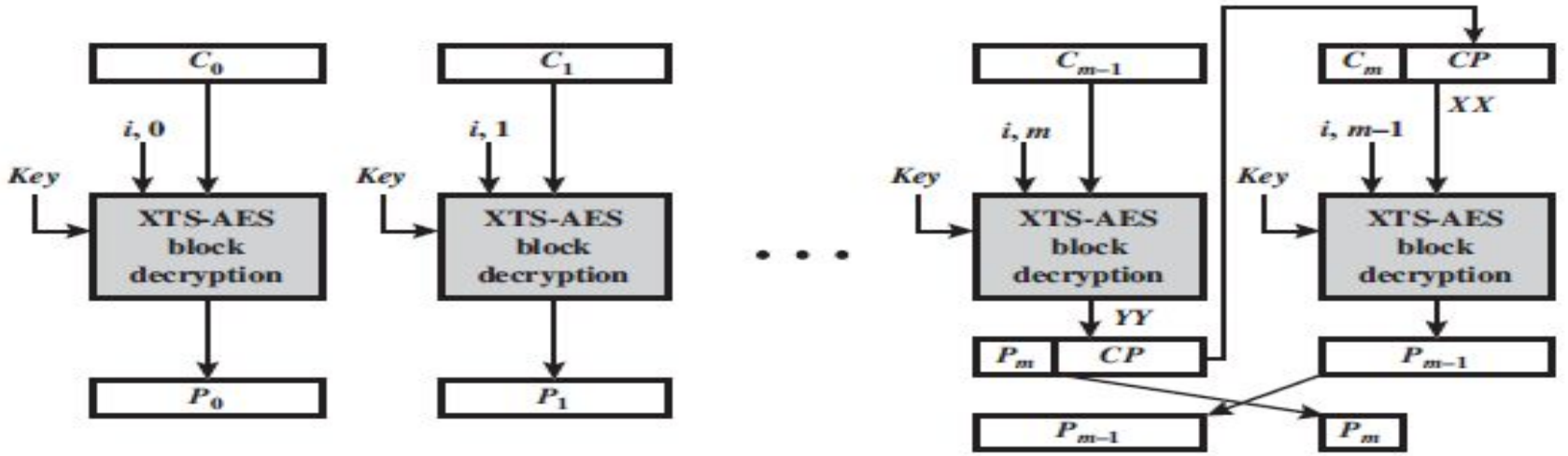
$$P = PP \oplus T = D(K_1, CC) \oplus T = D(K_1, C \oplus T) \oplus T$$

$$\begin{aligned} P &= D(K_1, C \oplus T) \oplus T \\ &= D(K_1, [E(K_1, P \oplus T) \oplus T] \oplus T) \oplus T \\ &= D(K_1, E(K_1, P \oplus T)) \oplus T \\ &= (P \oplus T) \oplus T = P \end{aligned}$$

Operation on a Sector



(a) Encryption



(b) Decryption

Operation on a Sector

XTS-AES mode with null final block	$C_j = \text{XTS-AES-blockEnc}(K, P_j, i, j) \quad j = 0, \ldots, m-1$
	$P_j = \text{XTS-AES-blockEnc}(K, C_j, i, j) \quad j = 0, \ldots, m-1$
XTS-AES mode with final block containing s bits	$C_j = \text{XTS-AES-blockEnc}(K, P_j, i, j) \quad j = 0, \ldots, m-2$ $XX = \text{XTS-AES-blockEnc}(K, P_{m-1}, i, m-1)$ $CP = \text{LSB}_{128-s}(XX)$ $YY = P_m \| CP$ $C_{m-1} = \text{XTS-AES-blockEnc}(K, YY, i, m)$ $C_m = \text{MSB}_s(XX)$
	$P_j = \text{XTS-AES-blockDec}(K, C_j, i, j) \quad j = 0, \ldots, m-2$ $YY = \text{XTS-AES-blockDec}(K, C_{m-1}, i, m-1)$ $CP = \text{LSB}_{128-s}(YY)$ $XX = C_m \| \text{CP}$ $P_{m-1} = \text{XTS-AES-blockDec}(K, XX, i, m)$ $P_m = \text{MSB}_s(YY)$

- Suitable for parallel operation like the CTR mode.