



FIT2093 INTRODUCTION TO CYBER SECURITY

COMMONWEALTH OF AUSTRALIA

Copyright Regulations 1969

WARNING

This material has been reproduced and communicated to you by or on behalf of Monash University pursuant to Part VB of the *Copyright Act 1968* (the Act).

The material in this communication may be subject to copyright under the Act. Any further reproduction or communication of this material by you may be the subject of copyright protection under the Act.

Do not remove this notice.



FIT2093 INTRODUCTION TO CYBER SECURITY

Lecture 4: Introduction to Cryptography

Unit Structure

- Introduction to security of
- Authentication
- Access Control
- **Fundamental concepts of cryptography**
- **Symmetric encryption techniques**
- Introduction to number theory
- Public key cryptography
- Integrity management
- Practical aspects of cyber security
- Hacking and countermeasures
- Database security
- IT risk management & Ethics and privacy

Previous Lecture

- **introduced access control principles**
 - subjects, objects, access rights
- **discretionary access controls**
 - access matrix, access control lists (ACLs), capability tickets
- **Mandatory access controls**
- **role-based access control**
- **Security Models**
 - Bell-Lapadula
- **UNIX Access Control mechanisms**

Outline

- **Definition of Cryptography**
- **Brief History of Cryptography**
- **Basic terms, notations and structure of cryptography**
- **Classical Cryptographic Techniques**
- **Modern Cryptographic Techniques**
- **Encryption and possible attacks**
- **Stream & Block Ciphers**
- **Modes of operation**

What we have seen so far?

- Two level of defence
- **Authentication** to gain access to the system
- **Access Control** (DAC by operating system and MAC by the organisation) to have rightful operation on the information resources such as files, programs, etc.
- **Third level** → to protect the contents of data resources
- In the following discussions we will use the term “**Messages**” to mean the contents of a data file or message exchanged between two parties (between users or user and program/host)

Cryptography

- **Derived from Greek words kryptos (hidden) and grapho (writing).**
- **The study of message secrecy.**



Cryptography – first set of terminology

- Original message → plain text
- Scrambled text → cipher text
- The aim of cryptography is to generate efficiently the cipher text from plain text and **vice-versa** if you know the secret scrambling information (usually referred as the **key**)



**Invertible or
reversible**

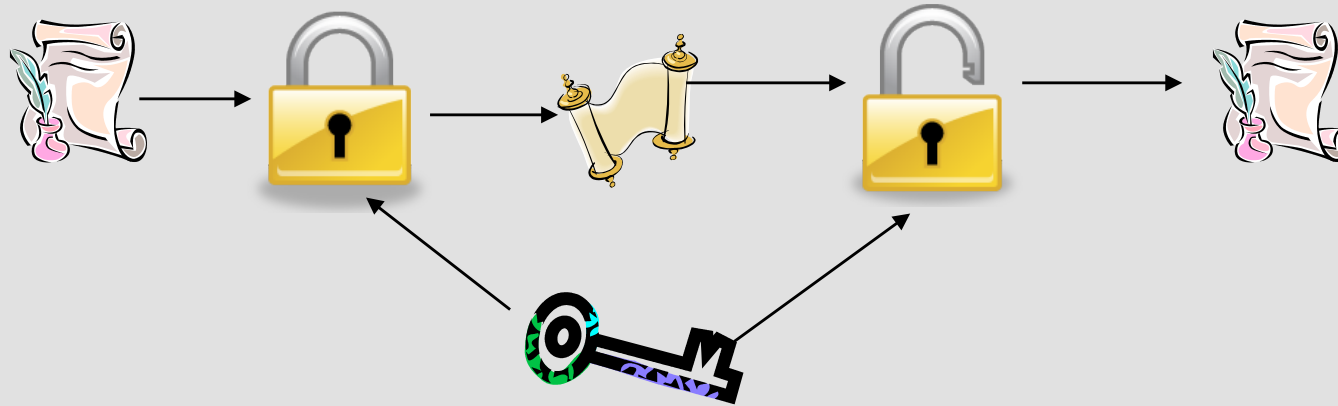
Basic Terminology

- **Cryptology**
 - Cryptography --- code designing
 - Cryptanalysis --- code breaking
- **Cryptologist:**
 - Cryptographer & cryptanalyst
- **Encryption/encipherment**
 - Scrambling plaintext data into unintelligible form to unauthorised parties
- **Decryption/decipherment**
 - Un-scrambling



History of Cryptography - Classic

- **Traditionally cryptography is used to provide message confidentiality.**



- **Symmetric Key (the same key is used on both sides – to encrypt and to decrypt)**

Two basic principles used to hide a message

- **Most of the classical ciphers are designed based on:**
 - **Substitution**
 - > Systematically substitute letters or a group of letters with other letters or a group of letters.
 - > Example:
 - change each letter with another letter two position down in the English alphabet sequence.
 - “message” => “oguucig”
 - **Transposition**
 - > Rearrange the position of the letters in the message according to some rules.
 - Move each letter in the message by 1 position.
 - » “message” => “emessag”
- **Heavily used in Classical Ciphers**

Classical Cipher – Caesar Cipher (1)

- The Caesar cipher is a substitution cipher, named after Julius Caesar.
- Operation principle:
each letter is translated into the letter *a fixed number of positions* after it in the alphabet table.
- The fixed number of positions is the **key** both for encryption and decryption.

Classical Cipher – Caesar Cipher (2)

K=3 – Shifted by three characters

Outer: plain text char

Inner: cipher text char



Classical Cipher – Caesar Cipher (3)

- **Caesar Cipher can be viewed as modulo operation**
- **Encryption:**
 - Let x to be the position of the letter to be replaced and n to be the number of positions to be shifted in order to find the replacement letter. i.e the key is the value of n .
 - $E_n(x) = (x+n) \bmod 26$ and substitute that character in that position.
- **Decryption:**
 - $D_n(x) = (x-n) \bmod 26$

An example

- **for a key $K=3$,**
 - plaintext letter: ABCDEF . . . PQRSTUVWXYZ
 - ciphertext letter: $\overset{\vee}{D}EFGHI . . . STUV\overset{\vee}{W}XYZABC$
- **Hence**
TREATY IMPOSSIBLE
- **is translated into**
WUHDWB LPSRVVLEOH

Breaking the Caesar cipher

- by trial-and error
- by using statistics on the usage of English alphabets
 - frequency distributions of letters

letter	percent
A	7.49%
B	1.29%
C	3.54%
D	3.62%
E	14.00%

.....

Classical Cipher – Rail Fence Cipher

- It is an example of a transposition cipher. The letters are placed in multiple number of lines. The letters start from line 1 position 1 to line 2 position 2, then, line 3 position 3, line 2 position 4, line 1 position 5, etc.
- The plaintext “WE ARE DISCOVERED FLEE AT ONCE”
W . . . E . . . C . . . R . . . L . . . T . . . E
. E . R . D . S . O . E . E . F . E . A . O . C .
. . A . . . I . . . V . . . D . . . E . . . N . .
- Message ciphertext: **wecrlte erdsoeefeaoc aivden**
- What is the key for this example of rail fence cipher?
- Rail fence of depth 3

[Wikipedia.org](https://en.wikipedia.org/wiki/Rail_fence_cipher)

Substitution

- Build a (mapping) table that maps a character with the replacement character
- Needs this table (in fact column 2) to encode/decode messages (assuming the alphabets in column 1 is in a known sequence)
- Different people can use different tables
- What property(ies) does this table need to satisfy?
- How many such mapping tables are possible for 26 upper-case English alphabets?
- 26! Mapping tables

A	D
B	G
C	I
D	B
E	J
F	C
G	H
H	A
I	E
J	F

**Invertible or
reversible**

Substitution (2)

- **is a permutation of a sequence of alphabets.**
- **Permutation table can be thought of as a key.**



Why combine these two operations?

- **Each method contributes its own strength to scrambling**
- **Hence the final strength is at least equal to the product of individual methods**

What properties should a scrambled message have? – Property 1

- **Confusion**

- seeks to make the relationship between the **statistics of the ciphertext** and the value of the **encryption key** as complex as possible
- This is achieved by complex substitution.
- Difficult to deduce the key

What property should a scrambled message have? – Property 2

- **Diffusion**

- The statistical structure of plaintext is dissipated in the long range statistics of the ciphertext
- Scrambling should spread the information from the plain text so that each the plain text digit affects many cipher text digits.
- This can be achieved by complex transposition

What property should a scrambled message have? – Property 3

- **Avalanche effect**

- is a property of any encryption algorithm such that a small change in either the **plaintext** or the **key** produces a significant change in the ciphertext.
- A change in one bit of plaintext or one bit of the key should produce a change in many bits of the ciphertext

What property should a scrambled message have? – **Properties (4)**

- From the view of information security we would like to have **unconditional security**
 - no matter how much computer power is available, the cipher cannot be broken since the ciphertext provides insufficient information to uniquely determine the corresponding plaintext
- From the view of computational efficiency we would like to have **computational security**
 - given limited computing resources (eg time needed for calculations is greater than useful age of information), the cipher cannot be broken
 - The cost of breaking the cipher exceeds the value of the encrypted information

Transposition and Substitution Table

- **Transposition** – plaintext is rearranged -**permutation**
- **Substitution** - Need to exchange and remember the substitution table
 - Table can be big, if the number of alphabets is large
 - Modulo arithmetic is expensive because of division
- Need **efficient** transposition method
- Need to **avoid** large substitution tables, but still need a method to substitute characters (or bits)



Let us pause for a minute and
talk about the coding of the
information that we want to
scramble

Information that we want to scramble is represented as

- **Can be thought of as a sequence of characters (includes alphabet, numbers, special characters)**
- **If it is expressed in English language, we can use all the keys in a normal key board**
 - how many distinct usable characters are there on a normal key board?
 - 128 (34 controls + 10 digits + 52 letters + 32 special characters)
- **Each keyboard character is represented by a 7bit/8 bit ASCII code inside a computer.**

Information that we want to scramble is represented as (2)

- Then a message of n characters is encoded into a sequence of 0 and 1s.
 - Here each character in the message is replaced by its corresponding ASCII code.
 - n characters in a message has $8 * n$ binary bits, assuming 8 bit ASCII coding.

Block

- **A long message can be broken down into equal sized blocks**
 - exception – the last block
 - pad with blank characters to make it equal size



Block (2)

- **Example:**
- **Consider a message of 100 characters and this message will be divided into blocks of 8 characters long, then**
- **block 1 \rightarrow char(1) - char(8), block 2 \rightarrow char(9)-char(16) ... block 13 \rightarrow char(97) – char(100)**
- **Each block has 64 bits (assuming that 8 bit ASCII coding is used per character) except the last block.**

Scrambling a message

- **Scramble each of the blocks of the message independently**
 - Block size is constant
 - Last block, if it is smaller, pad with special characters to make it the uniform size

Block or Symmetric Cipher

- **Problem of Encryption:** Given a block of **n** binary bits (say $n=64$ bits), apply
 - efficient **substitution** and **transposition** operations so that
 - the outputs satisfy the **confusion** and **diffusion** properties.
- **Known as block cipher.**



Back to FIT1001/FIT1031
One Slide Wisdom of
FIT1001/FIT1031/

Remember

- **that we are trying to scramble a block of bits (say, for example, 64 bits)**



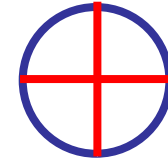
Transposition

- **Bit-shift (either to the left or to the right) operation can be performed in a single machine cycle.**
- **Fold operation – swapping half of the (left) bits with the other of half of the (right) bits.**
 - How efficiently can this operation be performed?
 - Which property (diffusion/confusion) is being achieved?

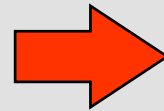
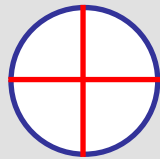
Substitution

- **Main property that a substitution needs to satisfy is**
 - the one-to-one mapping and
 - reversible.
- **Since the block is a sequence of bits, which binary operation is reversible and at the same time most efficient?**

Bit-wise exclusive-or (XOR)



101010111



011010101

0	\oplus	0	=	0
1	\oplus	1	=	0
0	\oplus	1	=	1
1	\oplus	0	=	1

110000010

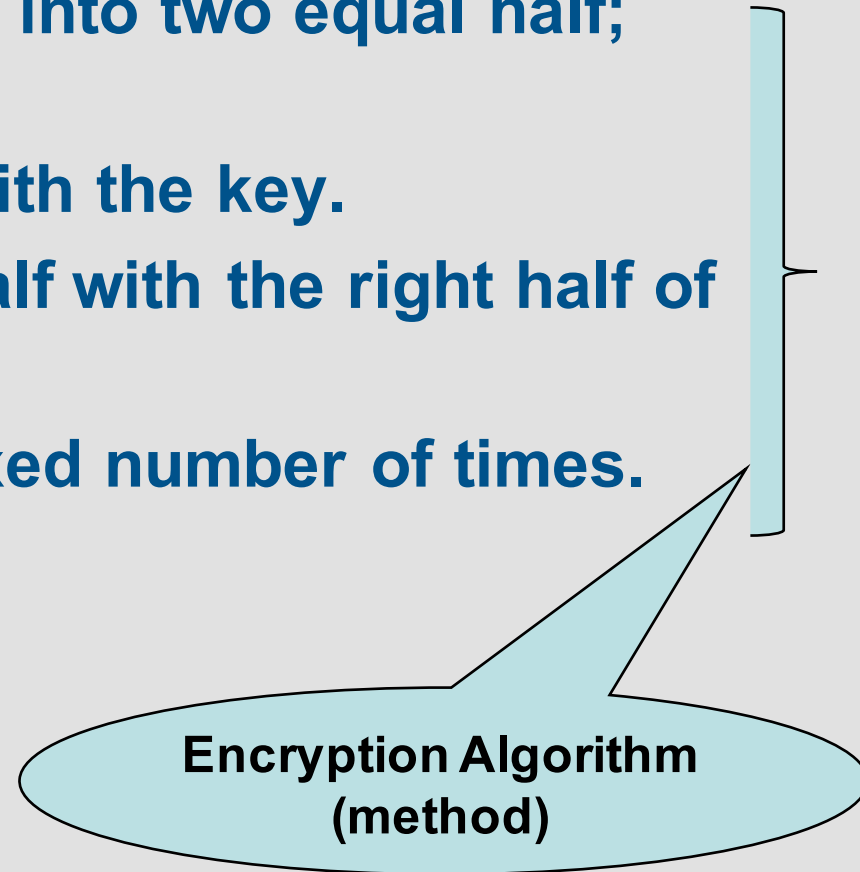


Substitution

- If you think of the first set of binary bits as the bits of a **data** block, and the second set of binary bits as the substitution table and the substitution operator is **XOR**.
- The second sequence of binary bits needs to be pre-designed.
 - Known as the key. Let us denote this by K .
 - The key length == the length of the block.

Putting all these for a simple cipher Algorithm can be

- **Step 1: Divide the block into two equal half; circular shift each half;**
- **Step 2: Perform XOR with the key.**
- **Step 3: Swap the left half with the right half of bits.**
- **Repeat steps 1 to 3 a fixed number of times.**



To decode the scrambled text back to its original message

- **Repeat the above steps in the reverse order!**
 - Bingo – you will have the original message.
 - An implementation of the previous slide idea is DES – Data Encryption Standard
 - its successor AES – Advance Encryption Standard.

Toy example of private key cryptography (TPC)

- Assume we have a **PKE** algorithm that uses **16-bits** block encryption and each 16-bit block of plaintext is encrypted separately.
- The key used to encrypt is based on a numerical value 0 to 7 and the place of the numerical value in the block, eg
 - key = [1,0] => numerical key value is 1 and this key should be applied to the first byte in the block.
 - key = [1,4] => apply numerical key 1 to the first byte and numerical key four to the second byte of the block.
- Assume that all 8 bits of a byte is used and key digits start from left to right.
- Encryption algorithm:
 - Step 1: Each plaintext block is first left circular shifted by the number of binary digits before the first non-zero digit of the key.
 - Step 2: Perform exclusive-or with the key starting from the first byte of the block, repeatedly to the end of the block. Decryption: do the reverse of encryption: the cipher-text is exclusive-ored and then right circular shifted.

0	\oplus	0	=	0
1	\oplus	1	=	0
0	\oplus	1	=	1
1	\oplus	0	=	1

\oplus : exclusive or

TPC in Action (1)

- **Plaintext “16” and key [1,6]**
- **Plaintext as blocks:**
 - 0000000100000110
- **Key [1,6]**
 - 0000000100000110
- **Encryption:**
 - Step 1: number of 0s before a non-zero digit in the **key** is 7, hence we left circular shift the bits in the message by 7 positions.
 - > scrambled message: 1000001100000000
 - Step 2: perform XOR operation on the scrambled message and the key.

MSG	1	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0
Key	0	0	0	0	0	0	0	1	0	0	0	0	0	1	1	0
CT	1	0	0	0	0	0	1	0	0	0	0	0	0	1	1	0



TPC in Action (2)

- **Decryption**

- Step 1: perform XOR between the ciphertext and the key.

1	0	0	0	0	0	1	0	0	0	0	0	0	1	1	0
0	0	0	0	0	0	0	1	0	0	0	0	0	1	1	0
1	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0

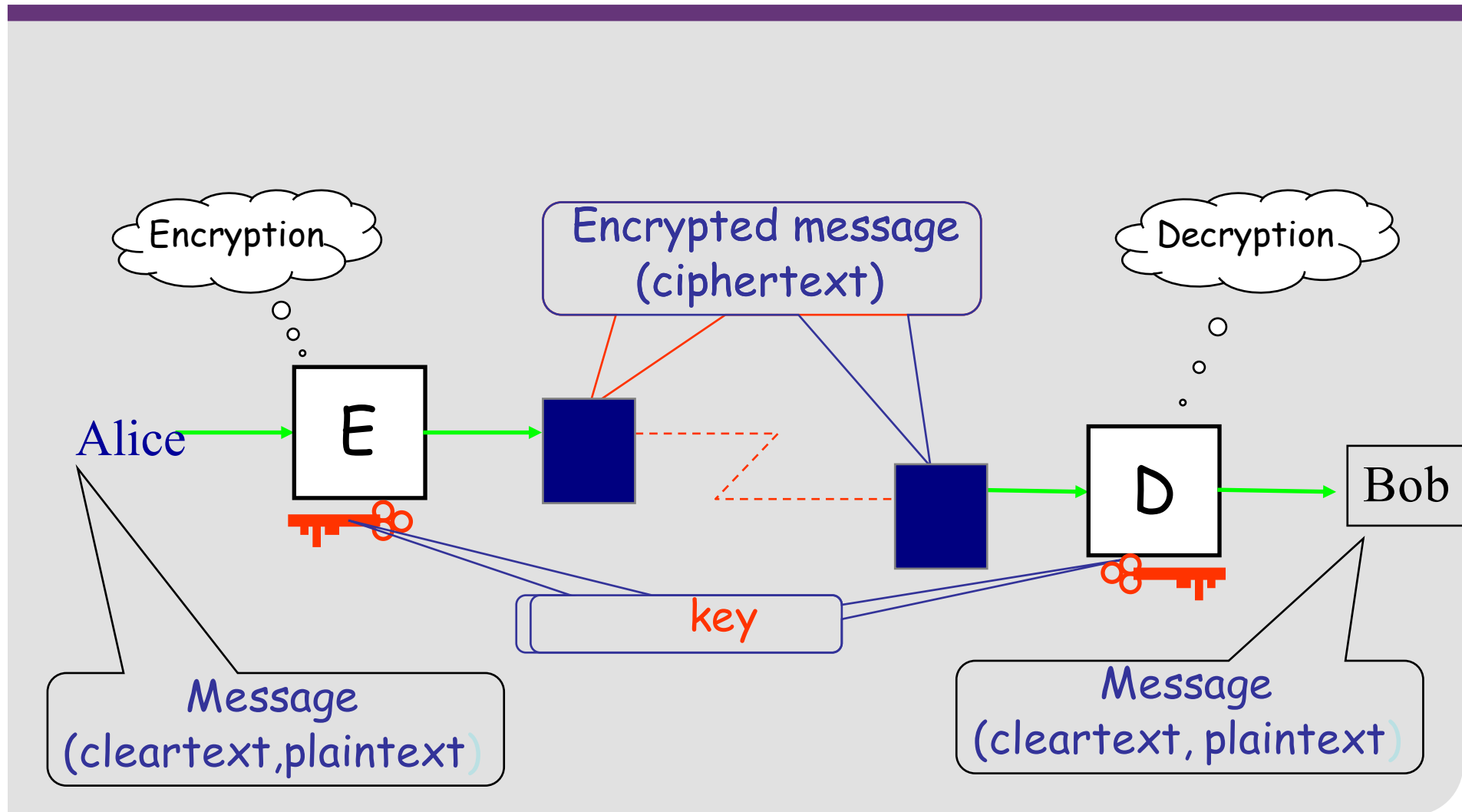
- Step 2: Circular shift the result of the XOR operation 7 bits to the right.
> 0000000100000110

Modern private key ciphers

- **DES (US, 1977) (3DES)**
 - key -- 56 bits, plaintext/ciphertext -- 64 bits
- **LOKI (ADFA, Australia, 1989)**
 - key, plaintext/ciphertext -- 64 bits
- **FEAL (NTT, Japan, 1990)**
 - key -- 128 bits, plaintext/ciphertext -- 64 bits
- **IDEA (Lai & Massey, Swiss, 1991)**
 - key -- 128 bits, plaintext/ciphertext -- 64 bits
- **SPEED (Y Zheng in 1996)**
 - Key/(plaintext/ciphertext) -- 48,64,80,...,256 bits
- **AES (Joan Daemen & Vincent Rijmen 2000)**
 - Key/(plaintext/ciphertext) -- 128, 192 and 256 bits



Symmetric key cipher



Symmetric Key Cryptography

- **A symmetric key cipher is composed of two algorithms**
 - encryption algorithm E
 - decryption algorithm D
- **The same key K is used for encryption & decryption**
- **K has to be distributed beforehand (how to communicate the key between two parties if they are separated?)**

Symmetric Key Cryptography - Notations

- **Encrypt a plaintext P using a key K & an encryption algorithm E**
 $C = E(K,P)$
- **Decrypt a ciphertext C using the same key K and the matching decryption algorithm D**
 $P = D(K,C)$
- **Note: $P = D(K,C) = D(K, E(K,P))$**



Feistel Cipher

- **Virtually all conventional block encryption algorithms, including DES (Data Encryption Standard) have a structure first described by Horst Feistel of IBM in 1973**

Classical Feistel Network

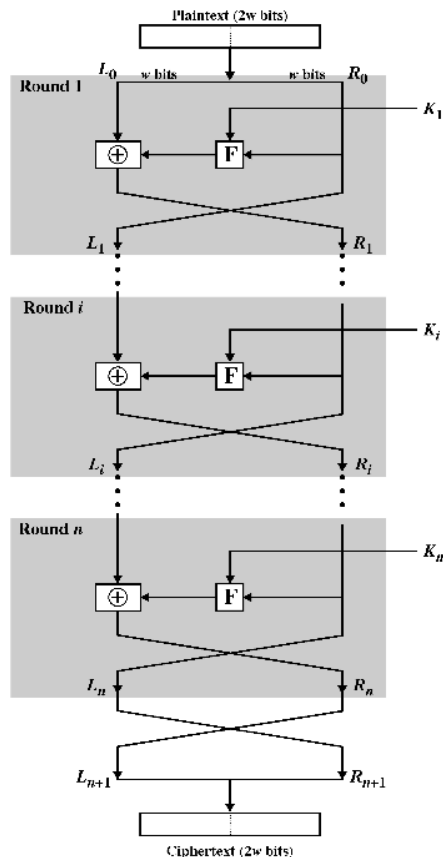


Figure 2.2 Classical Feistel Network

- For each round $i = 1, 2, \dots, n$, compute
- $L_i = R_{i-1}$
- $R_i = L_{i-1} \oplus f(R_{i-1}, K_i)$ where f is the round function and K_i is the sub-key
- decryption is accomplished via
- $R_{i-1} = L_i$
- $L_{i-1} = R_i \oplus f(R_{i-1}, K_i)$

Feistel Cipher Parameters

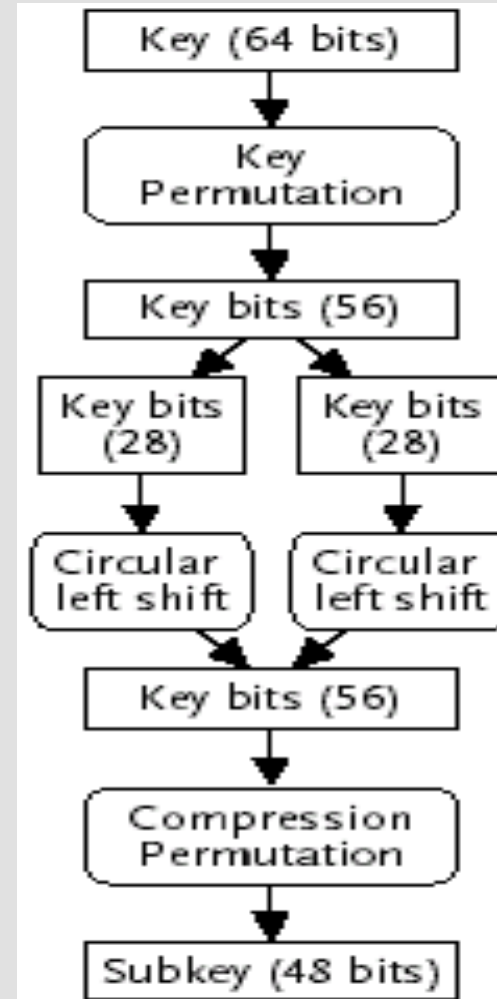
- **Fiestel Cipher depends on:**
 - **Block size:** larger block sizes mean greater security
 - **Key Size:** larger key size means greater security
 - **Number of rounds:** multiple rounds offer increasing security, typically 16 rounds
 - **Subkey generation algorithm:** greater complexity will lead to greater difficulty of cryptanalysis.
 - **Round function:** greater complexity means greater resistance to cryptanalysis

Symmetric Encryption Algorithms

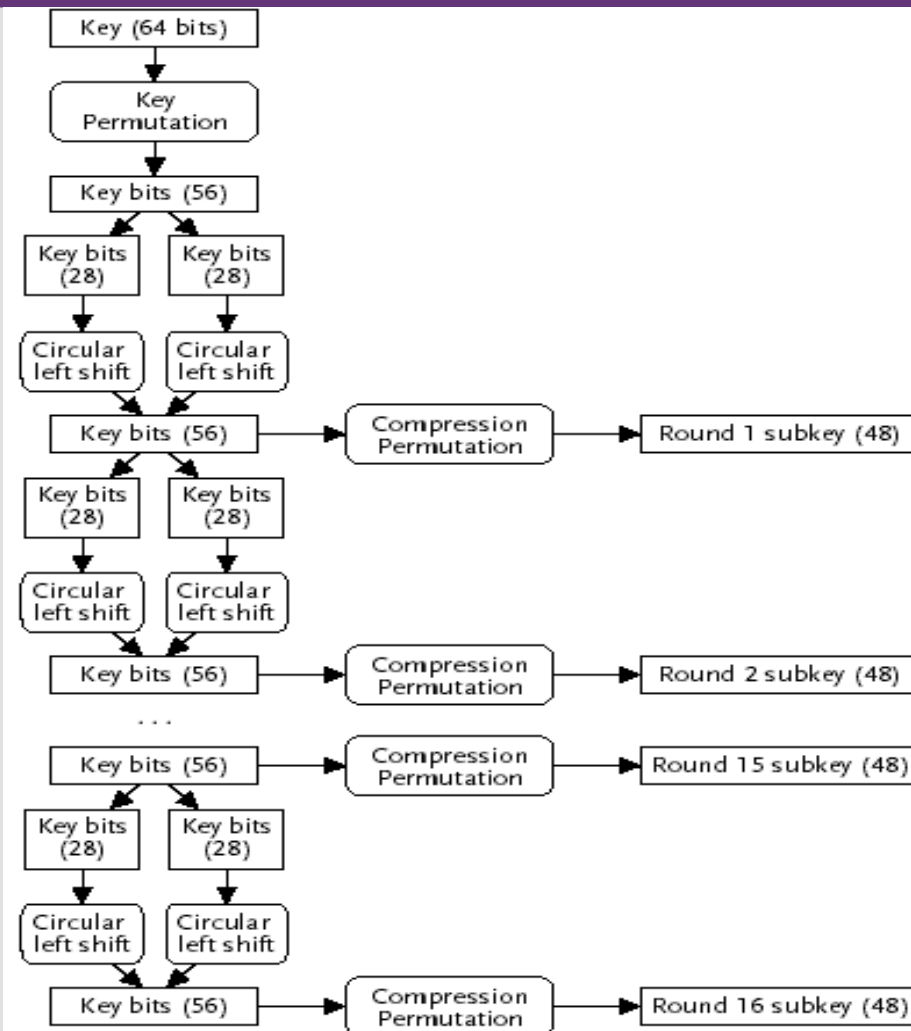
- **Data Encryption Standard (DES)**
 - The most widely used encryption scheme
 - The algorithm is referred to as the Data Encryption Algorithm (DEA)
 - DES is a block cipher
 - processed in 64-bit blocks
 - 56-bits key
 - 8 parity bits are stripped off from the full 64-bit key (8 characters)
 - 16 subkeys are generated for the 16 rounds

DES Subkey Generation - round # 1

- **drops 8 parity bits-
effective key size
becomes 56 bits**
- **permutes the bits and
divides into two 28-bits**
- **rotates the bits left by
single bit**
- **permutes and extracts 48
bits as a subkey**



DES Subkey Generation

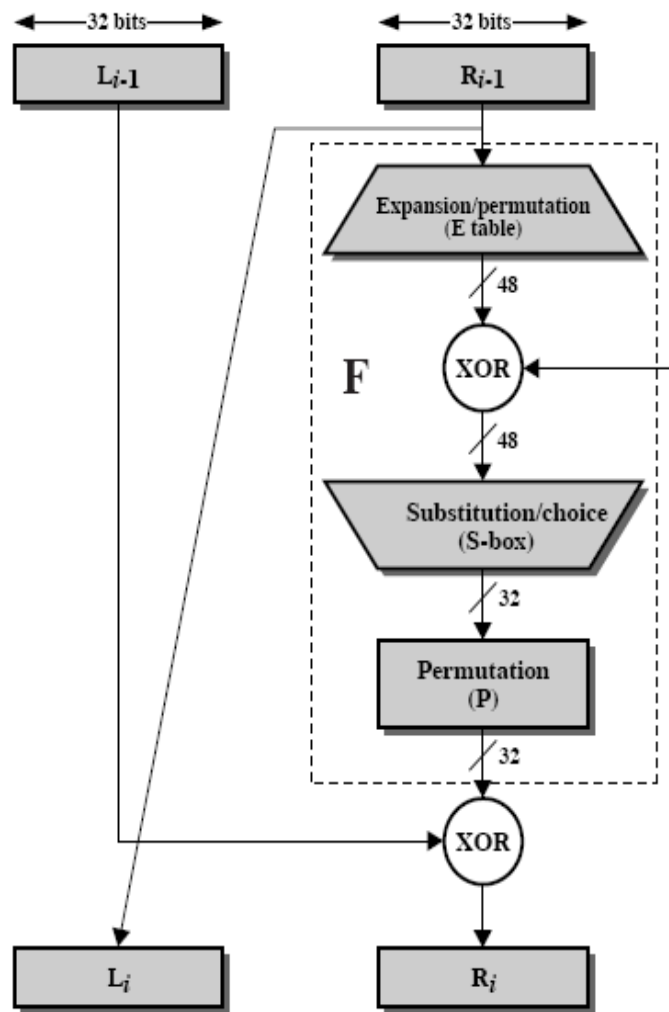


- One bit shift – round 1,2,9 and 16
- Two bit shift for the remaining rounds

DES Round

- Each of the sixteen rounds takes a 64-bit block as input and produces a 64-bit block as output
- The output from the initial permutation is the input to round one
- Round one's output is the input to round two
- Round two's output is the input to round three
- ...
- The output from round sixteen is the 64-bit block of ciphertext

Single DES Round



- Similar to Fiestel Cipher structure
- 64-bit plaintext is divided into two 32-bit blocks (L & R)
- L_i is the unchanged R_{i-1} (previous round)
- R_{i-1} goes through F function
 - E table-expanded to 48bits and permuted
 - 48 bits XORed with subkey K_i
 - Substitution produces 32-bit
 - > 8 S-boxes
 - > each takes 6 bits and produces 4 bits
 - > transformation is defined by substitution tables
 - > different substitution table for each S-box
 - Permutes the output of S-box
- R_i is L_{i-1} XORed with permuted output

DES Initial Permutation Table

Table 3.2 Permutation Tables for DES

(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



Brute Force Attack: Exhaustive Key Search

Key Size (bits)	Number of Alternative Keys	Time required at 1 encryption/ μs	Time required at 10^6 encryptions/ 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

Encryption Algorithms: Triple DES

- **Apply DES algorithm three times**
- **Use three keys and three executions of the DES algorithm (encrypt-decrypt-encrypt)**

$$C = E_{K3}[D_{K2}[E_{K1}[P]]]$$

C = ciphertext :

$E_K[X]$ = encryption of X using key K

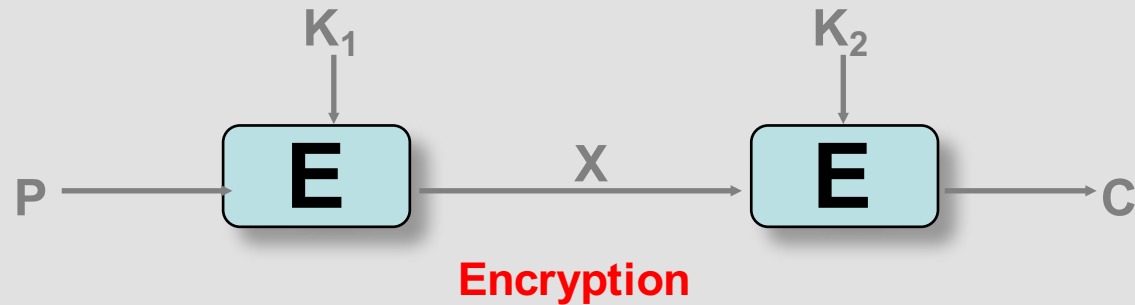
P = Plaintext :

$D_K[Y]$ = decryption of Y using key K

- **Effective key length of 168 bits = 3 * 56**
- **Decryption: same operation with keys reversed**

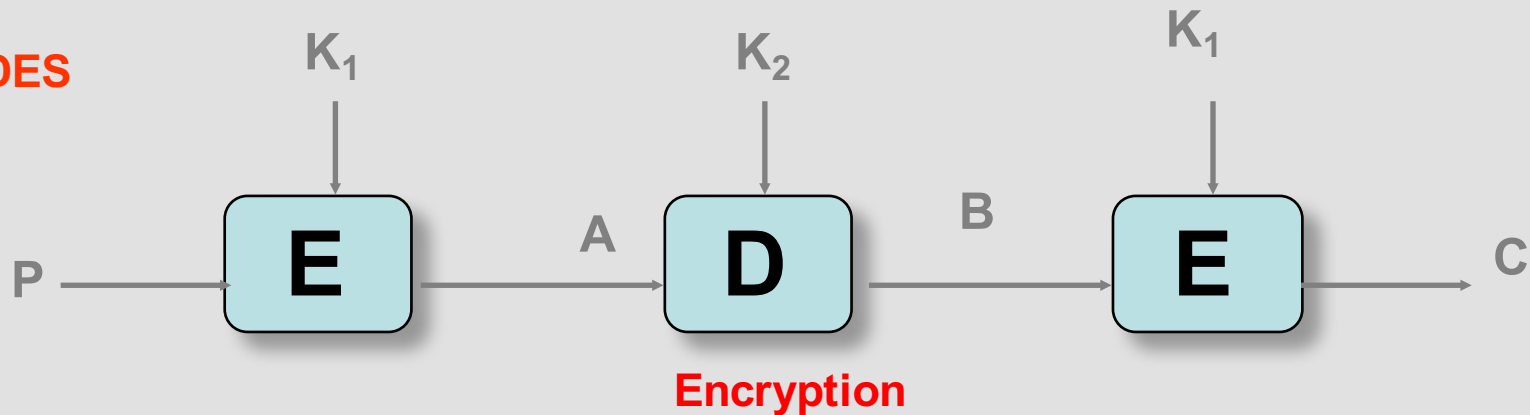
Double and Triple DES

Double DES

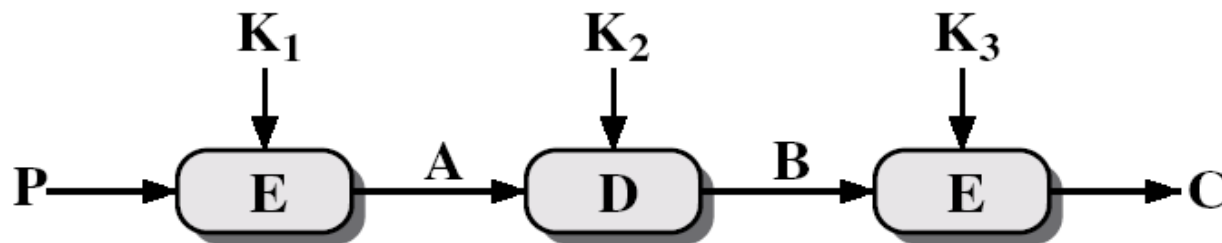


Encryption
Is double DES same as single DES with twice the length of the key?
Meet-in-the middle attack – what it is and is it possible here?

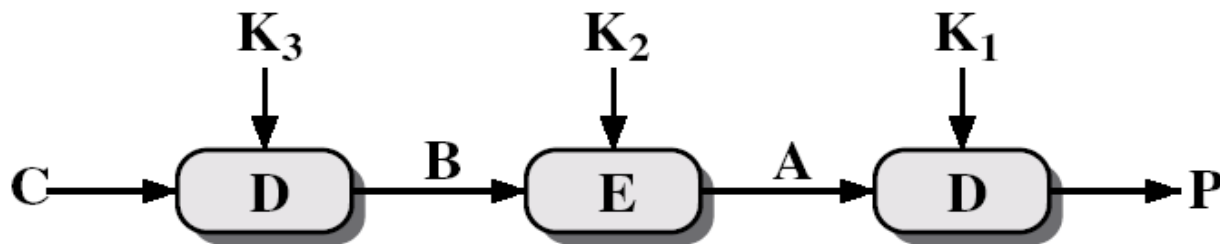
Triple DES



Encryption Algorithms: Triple DES...



(a) Encryption



(b) Decryption

Other Encryption Algorithms

- **International Data Encryption Algorithm (IDEA)**
 - Was developed to replace DES
 - 128-bit key, 64-bit block cipher
 - different round function and subkey generation from DES
 - Used in PGP
- **Blowfish**
 - DES-like algorithm: 64-bit cipher, 16 rounds
 - variable key length upto 448: 128-bit common
 - easy to implement, high execution speed, low memory requirement

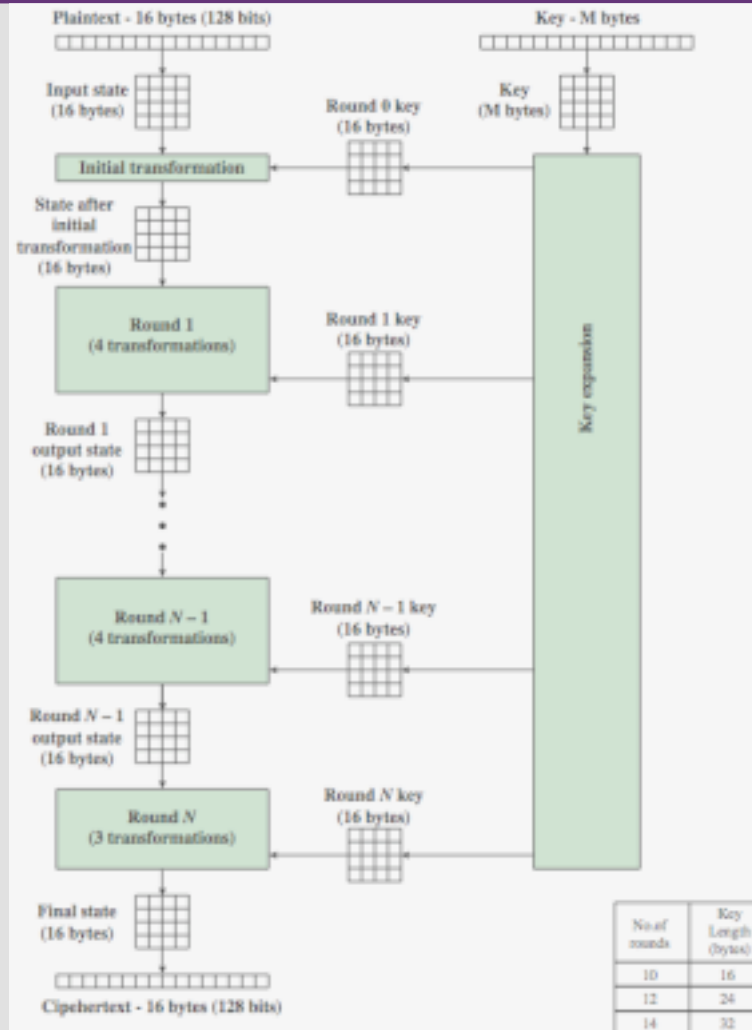
Other Encryption Algorithms...

- **Triple DES enjoyed widespread use for few years, but has severe drawbacks:**
 - sluggish in software
 - use 64-bit block size. A larger length is desirable for efficiency and security
- **In 1997, US National Institute of Standards and Technology (NIST) officially endorsed Advanced Encryption Standard (AES) to replace DES based on:**
 - security, computational efficiency, memory requirements, hardware and software suitability, flexibility

Encryption Algorithms (AES)

- **Was developed by Joan Daemen and Vincent Rijmen from Belgium – also known as Rijndael algorithm**
- **The algorithm uses**
 - 128-bit block cipher
 - supports three different key length: 128, 192 and 256 bits
 - support 10 to 14 rounds
 - Offers flexibility
 - efficient implementation in both software and hardware

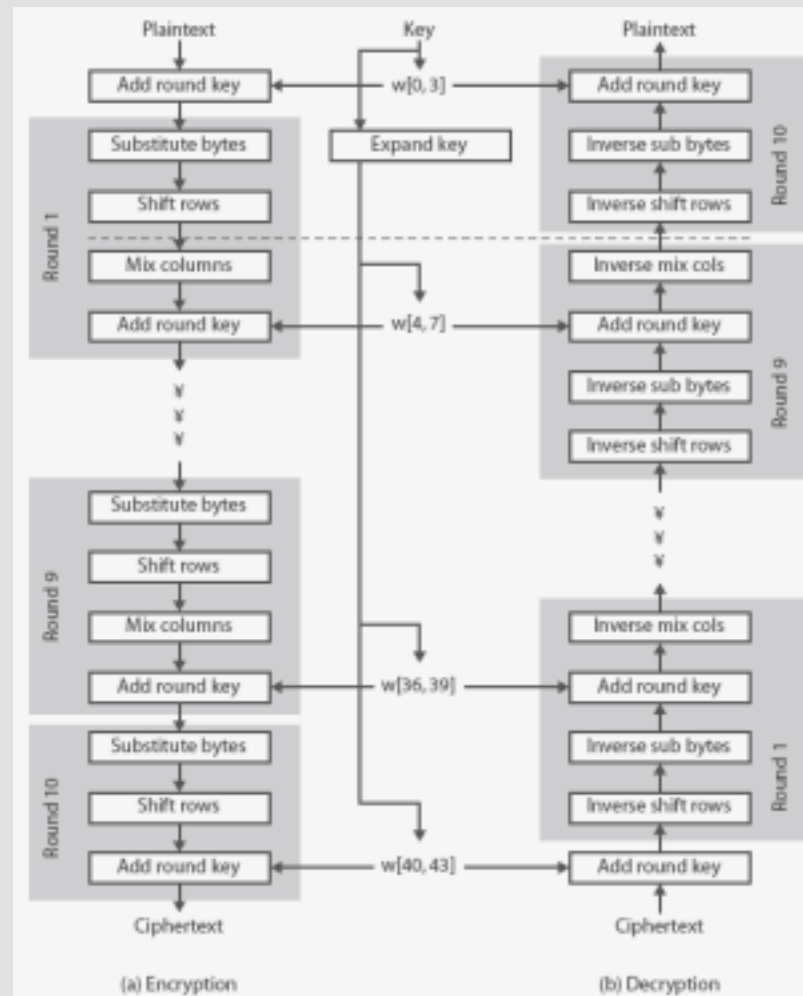
AES Encryption Process



AES CIPHER

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

AES Encryption and Decryption



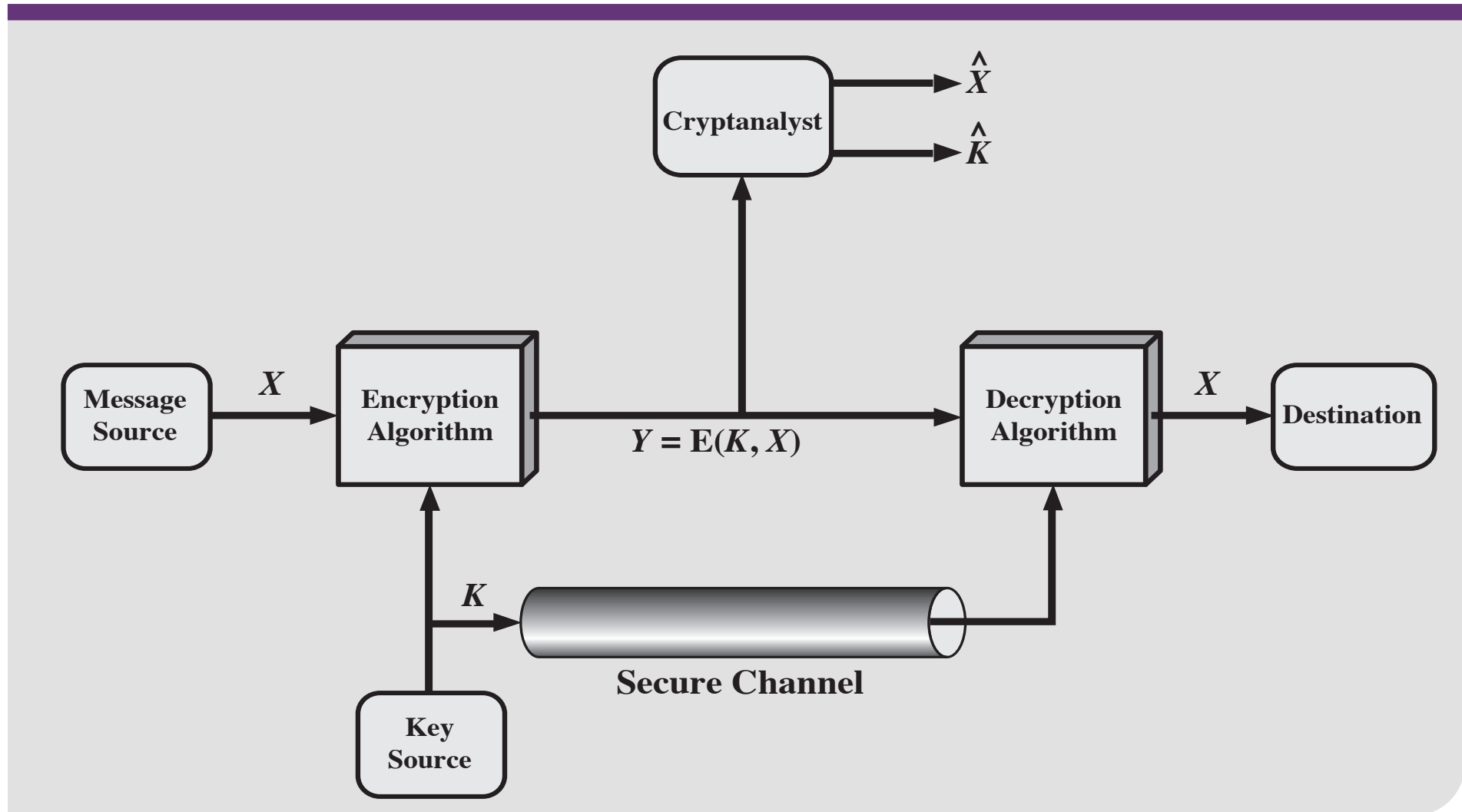
Implementation Aspects

- **can be efficiently implement on 8-bit CPU**
 - byte substitution works on bytes using a table of 256 entries
 - shift rows is simple byte shift
 - add round key works on byte XOR's
 - mix columns requires matrix multiply in $GF(2^8)$ which works on byte values, can be simplified to use table lookups & byte XOR's

The Attack

- **We consider confidentiality as the goal:**
 - Alice and Bob are Friends
 - Marvin is a rival
 - Alice wants to send secret messages (M_1, M_2, \dots) to Bob over the Internet
 - Rival Marvin wants to read the messages (M_1, M_2, \dots) - Alice and Bob want to prevent this!
 - Assumption: The network is OPEN: Marvin is able to eavesdrop and read all data sent from Alice to Bob.
 - Consequence: Alice must not send messages (M_1, M_2, \dots) directly – they must be “scrambled” or encrypted using a ‘secret code’ unknown to Marvin but known to Bob.

Model of Symmetric Cryptosystem



Cryptanalysis and Brute-Force Attack

Cryptanalysis

- Attack relies on the nature of the algorithm plus some knowledge of the general characteristics of the plaintext
- Attack exploits the characteristics of the algorithm to attempt to deduce a specific plaintext or to deduce the key being used

Brute-force attack

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




Brute-Force Attack

On average, half of all possible keys must be tried to achieve success



Involves trying every possible key until an intelligible translation of the ciphertext into plaintext is obtained



To supplement the brute-force approach, some degree of knowledge about the expected plaintext is needed, and some means of automatically distinguishing plaintext from garble is also needed

4 types of cryptanalysis

- **Depending on what a cryptanalyst has to work with, attacks can be classified into**
 - ciphertext only attack
 - known plaintext attack
 - chosen plaintext attack
 - chosen ciphertext attack (most severe)



4 types of attacks(1)

- **Ciphertext only attack**
 - The code breaker only has access to some samples of ciphertext.
- **Known plaintext attack**
 - The code breaker knows some samples of ciphertext with their corresponding plaintext.



4 types of attacks (2)

- **Chosen plaintext attacks**
 - can feed encryption algorithm with plaintexts and obtain the matching ciphertexts
- **Chosen ciphertext attack**
 - can feed decryption algorithm with ciphertexts and obtain the matching plaintexts



Types of Attacks on Encrypted Messages

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



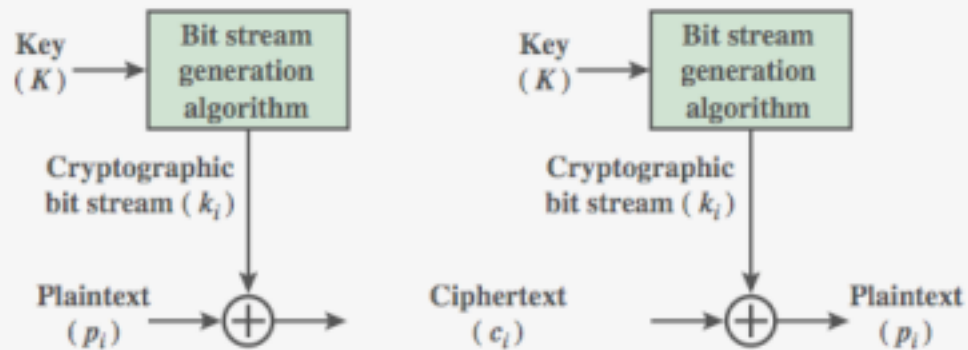
Stream Ciphers

Block & Stream Ciphers

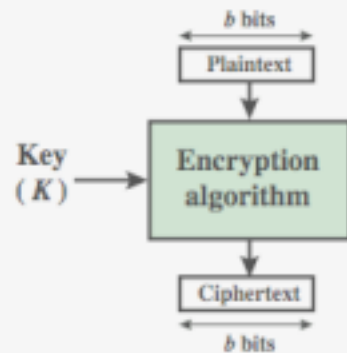
- **2 types of private key ciphers**
 - Block ciphers
 - encrypting block by block
 - > DES, AES, FEAL, IDEA, SPEED, ...
 - Stream ciphers (also called Vernam ciphers)
 - encrypting byte or bit by bit



Block vs Stream Ciphers



(a) Stream Cipher Using Algorithmic Bit Stream Generator

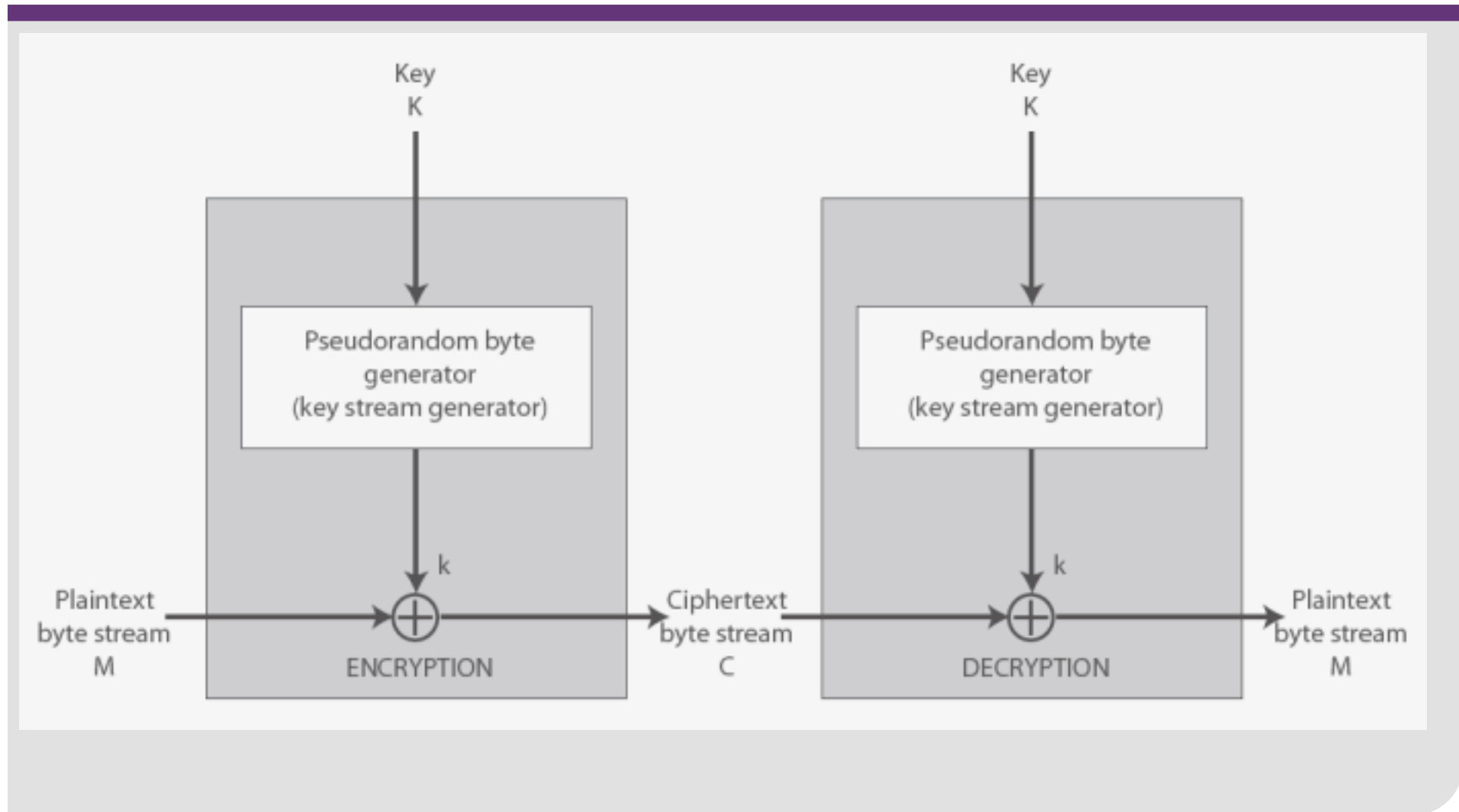


(b) Block Cipher

Stream Ciphers

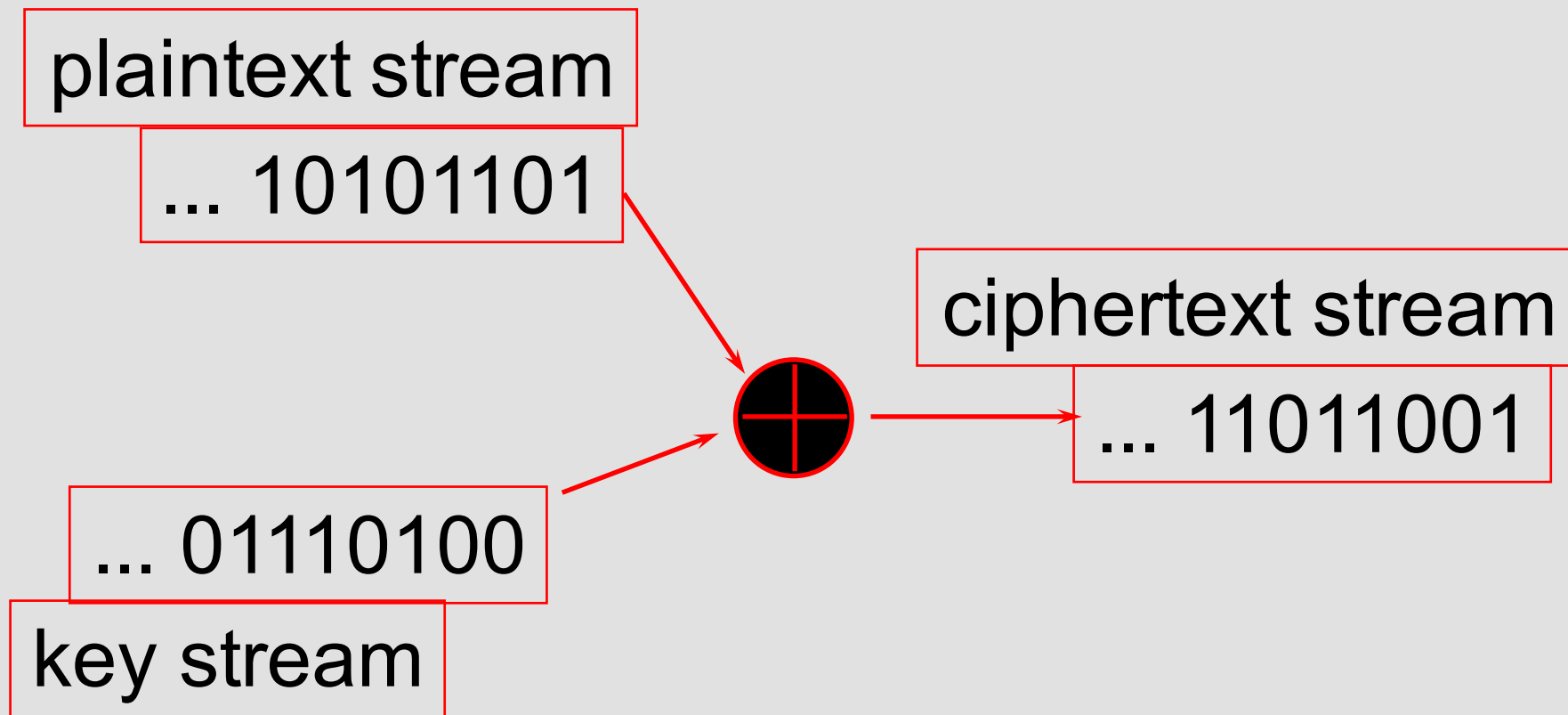
- process message one byte (or bit) at a time (as a stream)
- have a pseudo random keystream
- combined (XOR) with plaintext bit by bit
- randomness of **stream key** completely destroys statistical properties in the message
 - $C_i = M_i \text{ XOR } \text{StreamKey}_i$
- but must never reuse stream key
 - otherwise can recover messages (cf book cipher)

Stream Cipher Structure



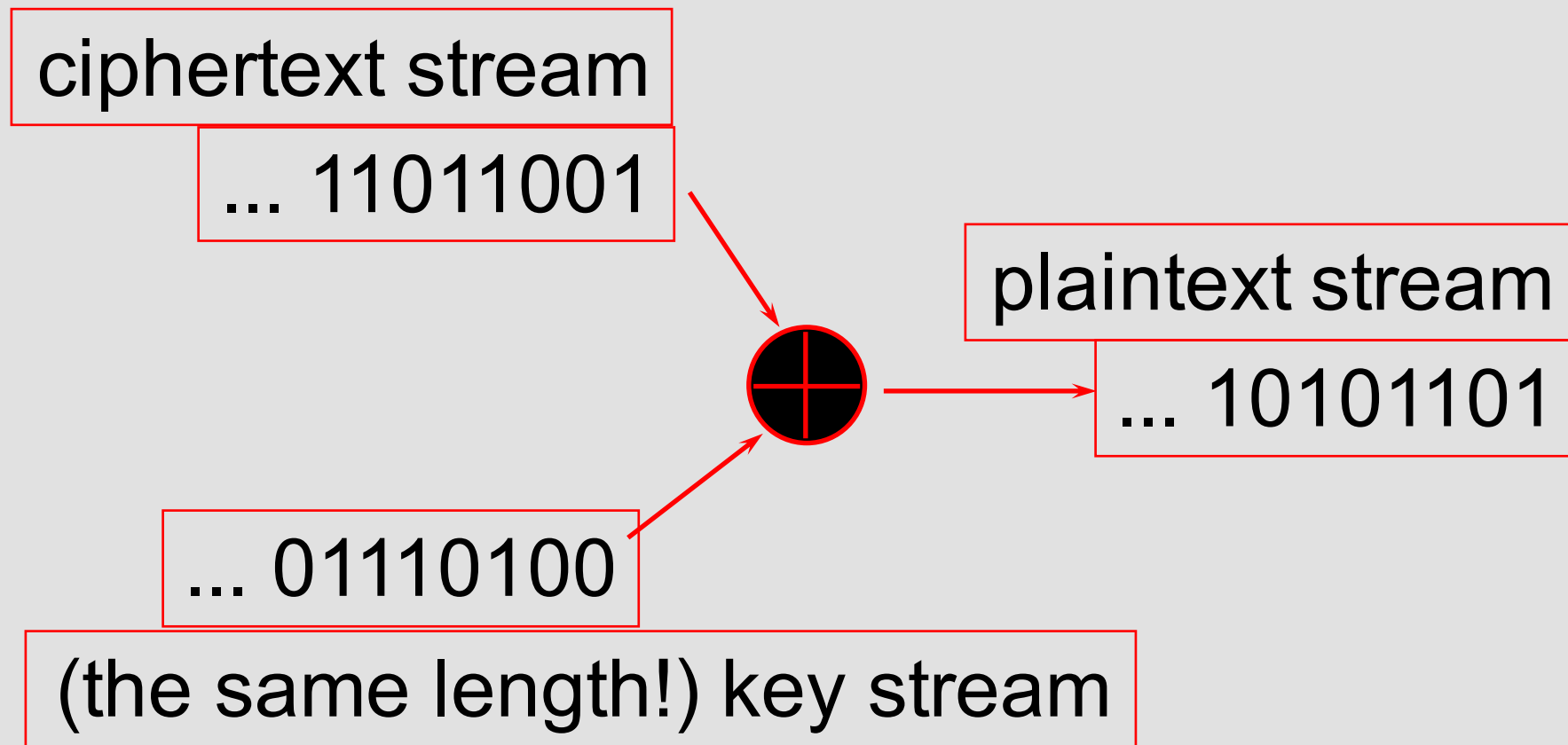
Stream ciphers

Encryption



Stream ciphers

Decryption



Never re-use a key

- **A key should never be re-used to encrypt two or more messages !**

- if $C_1 = M_1 \oplus K$, and
 $C_2 = M_2 \oplus K$

- then

- > if C_1, C_2 and M_1 are known, M_2 becomes known as well:

- $K = C_1 \oplus M_1$, so we have

- $M_2 = C_2 \oplus K$.

- > even if M_1 or M_2 is not known,

- $M_1 \oplus M_2 = C_1 \oplus C_2$

- may leak information on M_1 & M_2 .

- Key is One-time pad

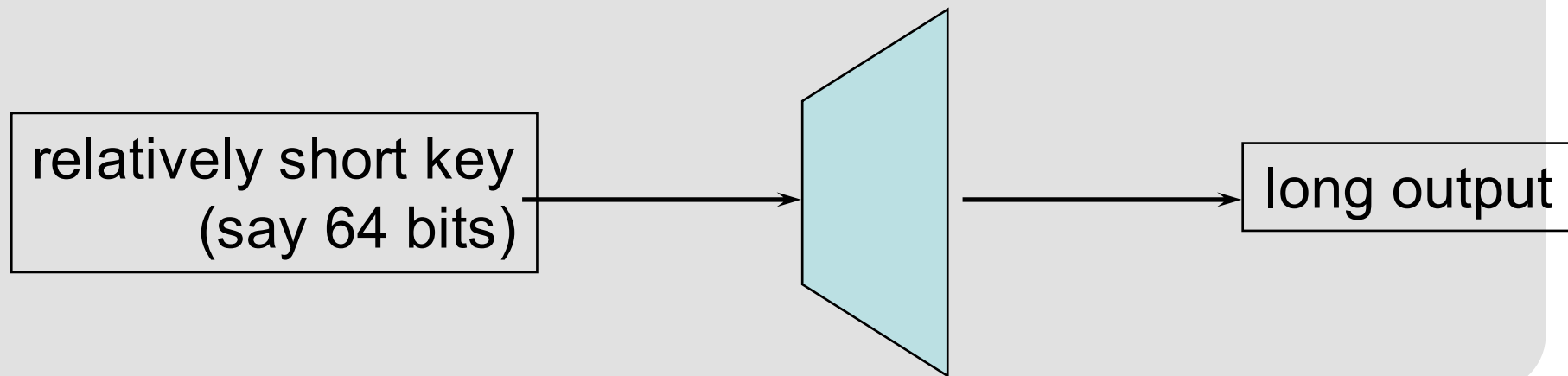
The too-long-key problem

- A big problem with a stream cipher is that a key must be as long as a plaintext message !
- Using a pseudo-random number generator (PRNG) can solve the problem !



Pseudo-random number generators

- A pseudo-random number generator (PRNG) is an algorithm that
 - expands
 - > a **random but relatively short** key, say of 64 bits, into
 - > a **long** bit sequence that preserves the randomness property of the original short key

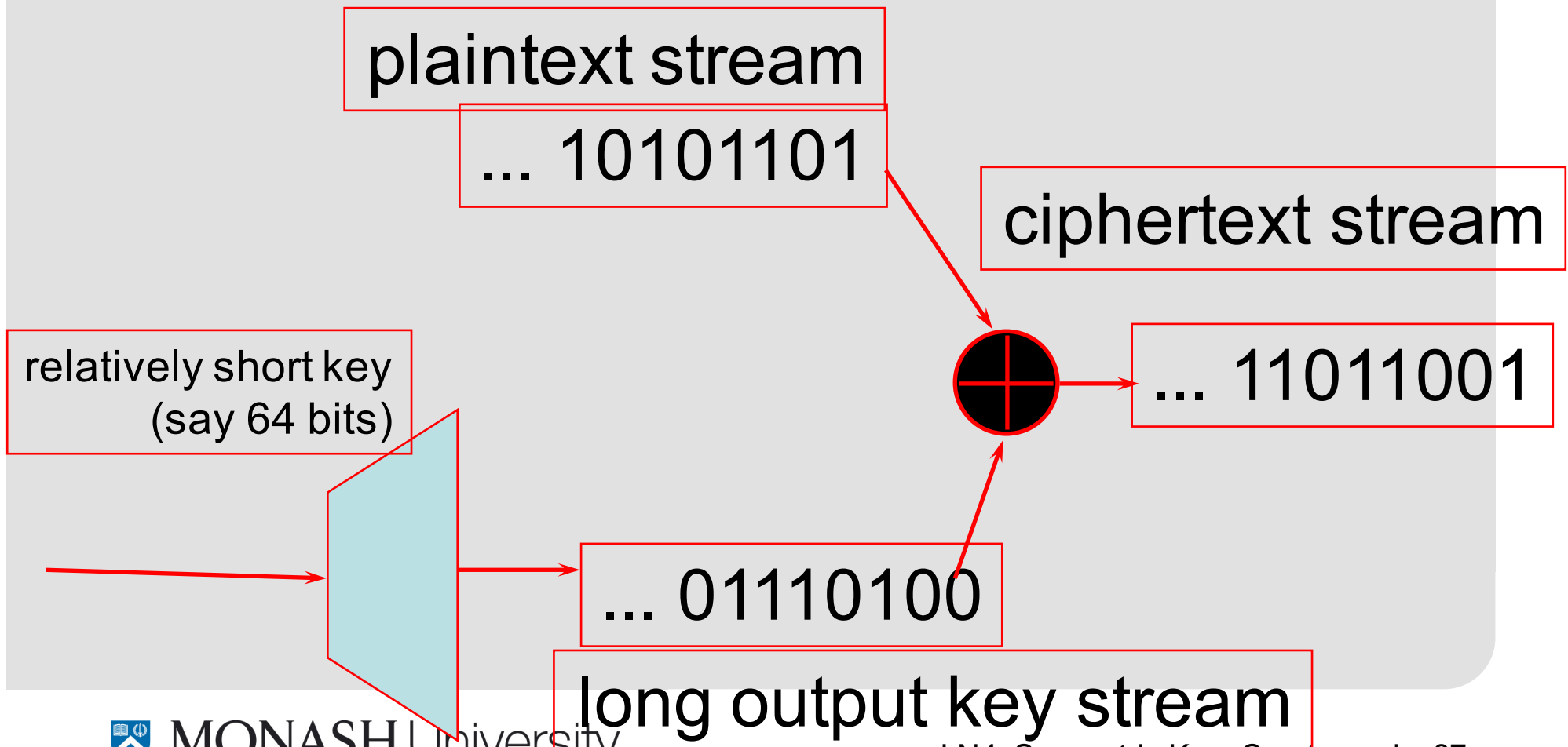


Other generators

- **A private key block cipher, such DES, acts as a pseudo-random number generator, when it is used in the Output Feedback Mode (OFB).**
- **There are many other types of pseudo-random number generators:**
 - based on Linear Feedback Shift Registers (LFSR)
 - based on Non- Linear Feedback Shift Registers (NLFSR)
 -

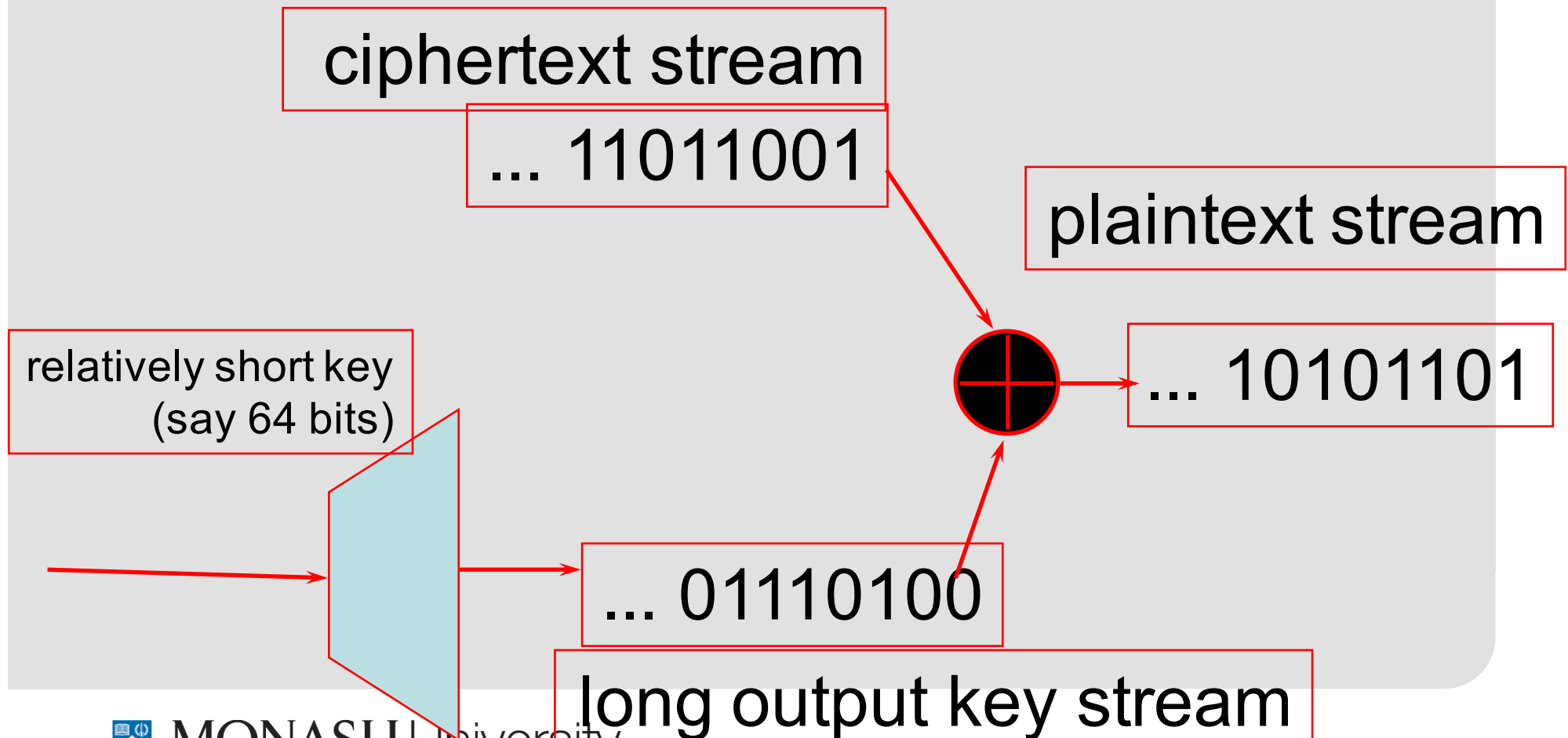
Stream ciphers with PRNG

Encryption



Stream ciphers with PRNG

Decryption



Stream Cipher Properties

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

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)
- key forms random permutation of all 8-bit values
- uses that permutation to scramble input info processed a byte at a time



Modes of Operation

Modes of Operation

- **block ciphers process data in blocks**
 - e.g. 64-bits (DES, 3DES) or 128-bits (AES)
- **for longer messages must break up**
 - and possibly pad end to block size multiple
- **have 5 five modes of operation for application of block cipher**
 - defined in NIST SP 800-38A
 - modes are: ECB, CBC, CFB, OFB, CTR

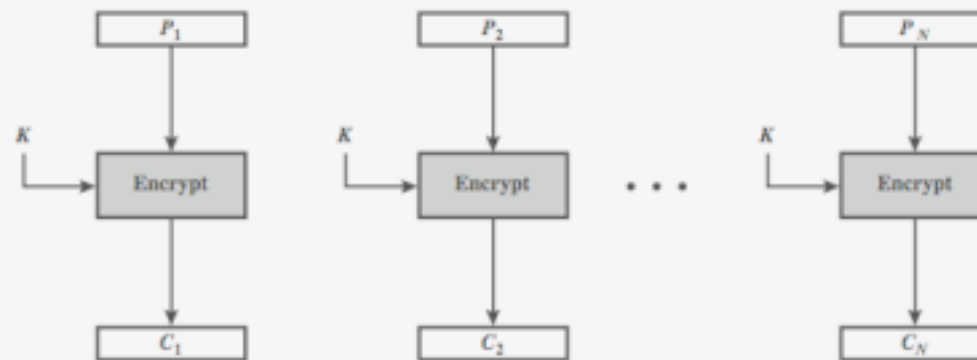
Electronic Codebook Book (ECB)

- **message is broken into independent blocks which are encrypted**
- **each block is a value which is substituted, like a codebook, hence name**
- **each block is encoded independently of the other blocks**

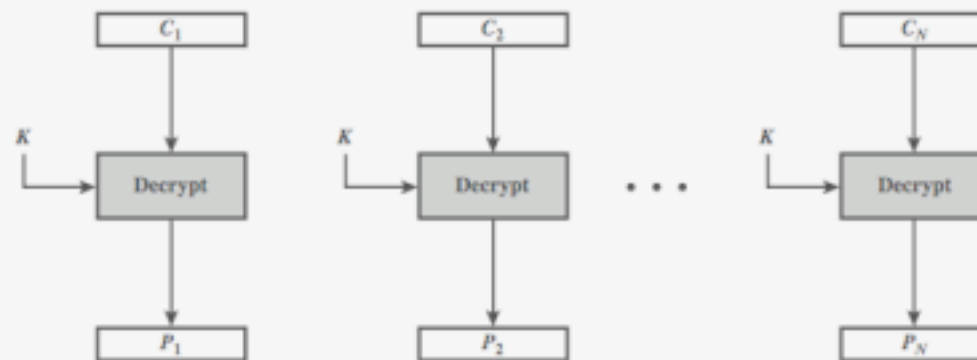
$$C_i = E_K(P_i)$$

- **uses: secure transmission of single values**

Electronic Codebook Book (ECB)



(a) Encryption



(b) Decryption



Advantages and Limitations of ECB

- **message repetitions may show in ciphertext**
 - if aligned with message block
 - particularly with data such graphics
 - or with messages that change very little, which become a code-book analysis problem
- **weakness is due to the encrypted message blocks being independent**
- **main use is sending a few blocks of data**

Cipher Block Chaining (CBC)

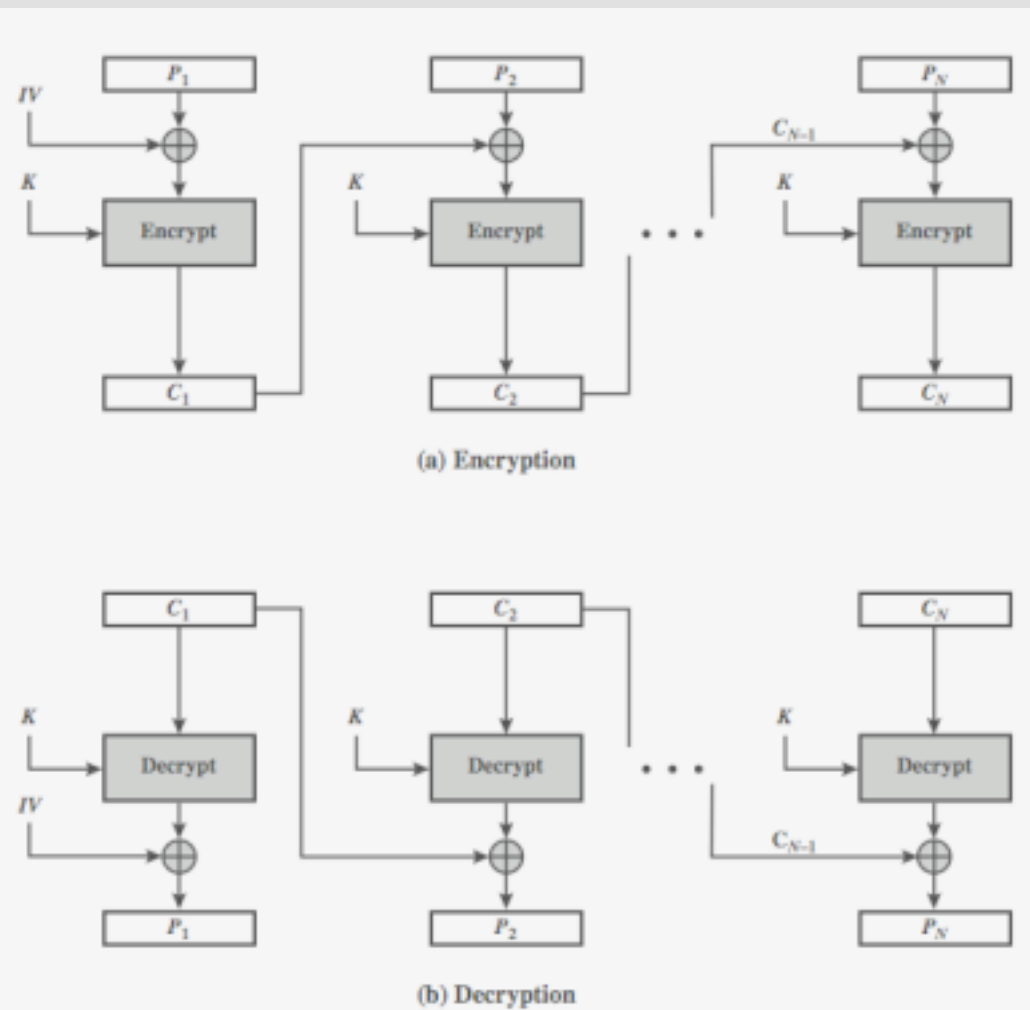
- **message is broken into blocks**
- **linked together in encryption operation**
- **each previous cipher blocks is chained with current plaintext block, hence name**
- **use Initial Vector (IV) to start process**

$$C_i = E_K(P_i \text{ XOR } C_{i-1})$$

$$C_{-1} = IV$$

- **uses: bulk data encryption, authentication**

Cipher Block Chaining (CBC)



Message Padding

- **at end of message must handle a possible last short block**
 - which is not as large as blocksize of cipher
 - pad either with known non-data value (eg nulls)
 - or pad last block along with count of pad size
 - > eg. [b1 b2 b3 0 0 0 0 5]
 - > means have 3 data bytes, then 5 bytes pad+count
 - this may require an extra entire block over those in message
- **there are other, more esoteric modes, which avoid the need for an extra block**

Advantages and Limitations of CBC

- **a ciphertext block depends on all blocks before it**
- **any change to a block affects all following ciphertext blocks**
- **need Initialization Vector (IV)**
 - which must be known to sender & receiver
 - if sent in clear, attacker can change bits of first block, and change IV to compensate
 - hence IV must either be a fixed value (as in EFTPOS)
 - or must be sent encrypted in ECB mode before rest of message

Stream Modes of Operation

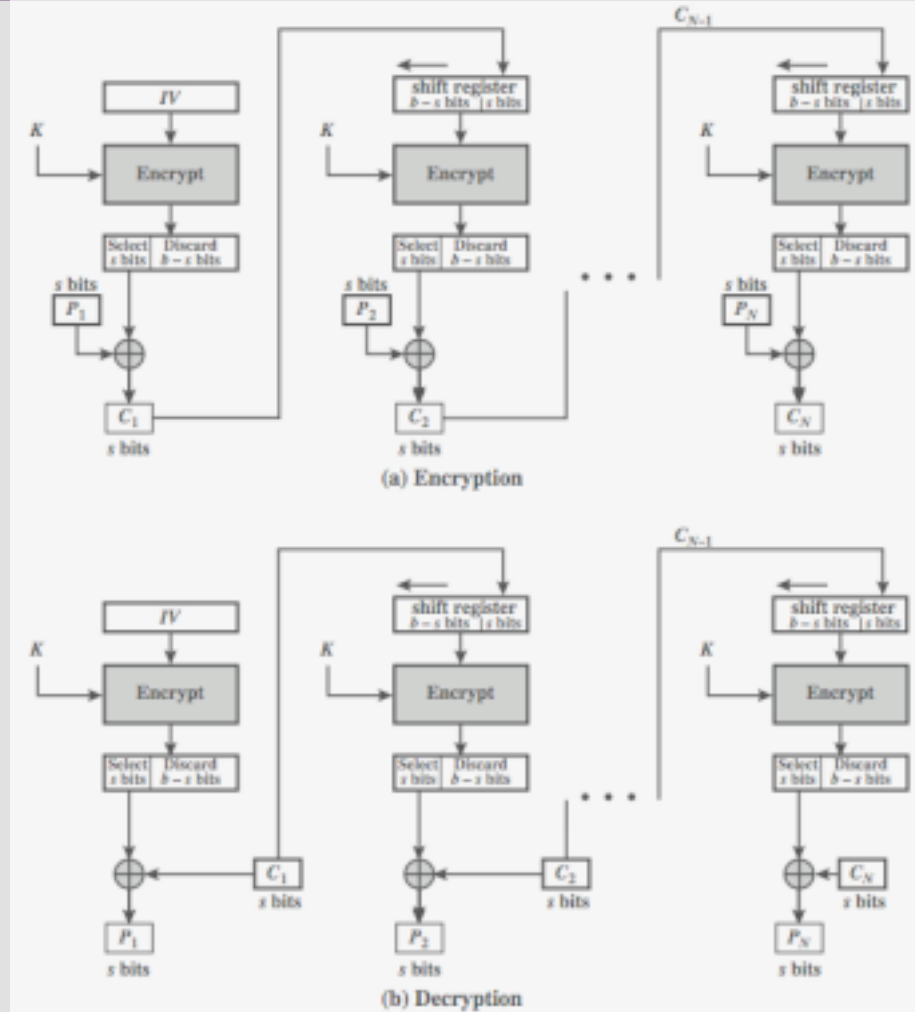
- **block modes encrypt entire block**
- **may need to operate on smaller units**
 - real time data
- **convert block cipher into stream cipher**
 - cipher feedback (CFB) mode
 - output feedback (OFB) mode
 - counter (CTR) mode
- **use block cipher as some form of pseudo-random number generator**

Cipher FeedBack (CFB)

- **message is treated as a stream of bits**
- **added to the output of the block cipher**
- **result is a feed back for next stage (hence name)**
- **standard allows any number of bit (1,8, 64 or 128 etc) to be feed back**
 - denoted CFB-1, CFB-8, CFB-64, CFB-128 etc
- **most efficient to use all bits in block (64 or 128)**
$$C_i = P_i \text{ XOR } \text{DES}_{K1}(C_{i-1})$$
$$C_{-1} = \text{IV}$$
- **uses: stream data encryption, authentication**

Cipher FeedBack (CFB)

s-bit Cipher FeedBack (CFB-s)



Advantages and Limitations of CFB

- appropriate when data arrives in bits/bytes
- most common stream mode
- limitation is need to stall while do block encryption after every n-bits
- note that the block cipher is used in **encryption** mode at **both** ends
- errors propagate for several blocks after the error

Output FeedBack (OFB)

- message is treated as a stream of bits
- A private key block cipher, such DES, acts as a pseudo-random number generator, when it is used in the Output Feedback Mode (OFB).
- output of block cipher is added to message
- (Block cipher) output is then fed back (hence name)
- feedback is independent of message
- can be computed in advance

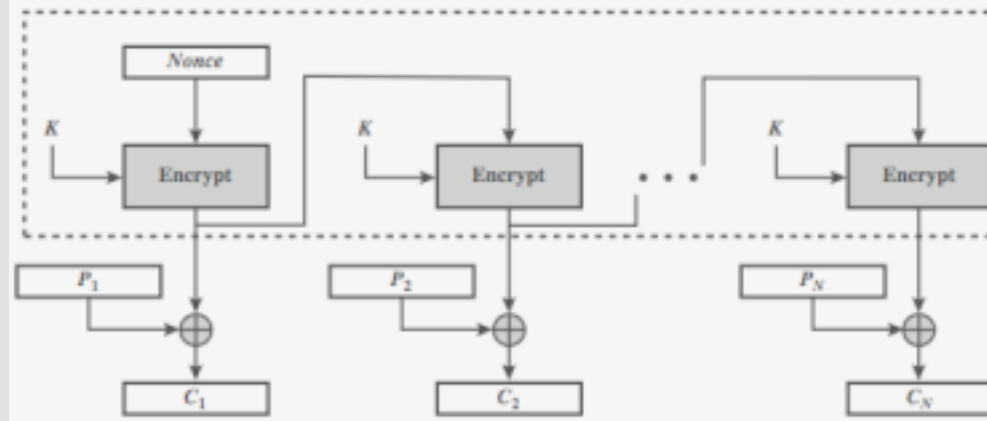
$$C_i = P_i \text{ XOR } O_i$$

$$O_i = \text{DES}_{K1}(O_{i-1})$$

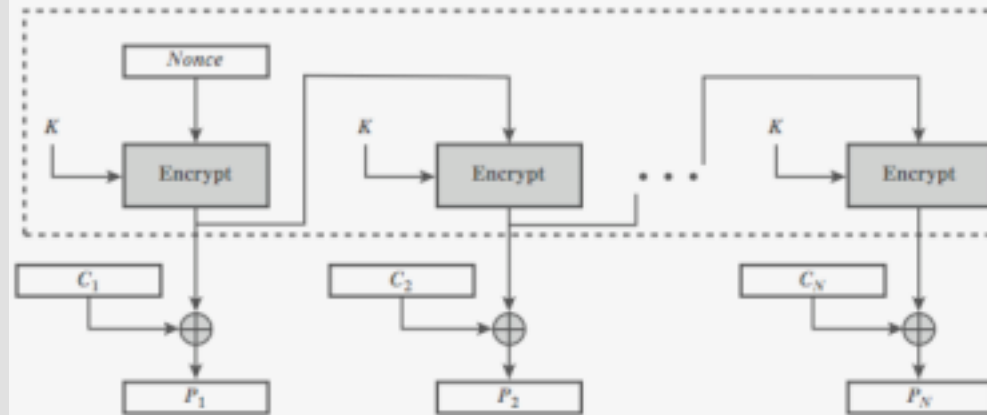
$$O_{-1} = \text{IV}$$

- uses: stream encryption on noisy channels

Output FeedBack (OFB)



(a) Encryption



(b) Decryption



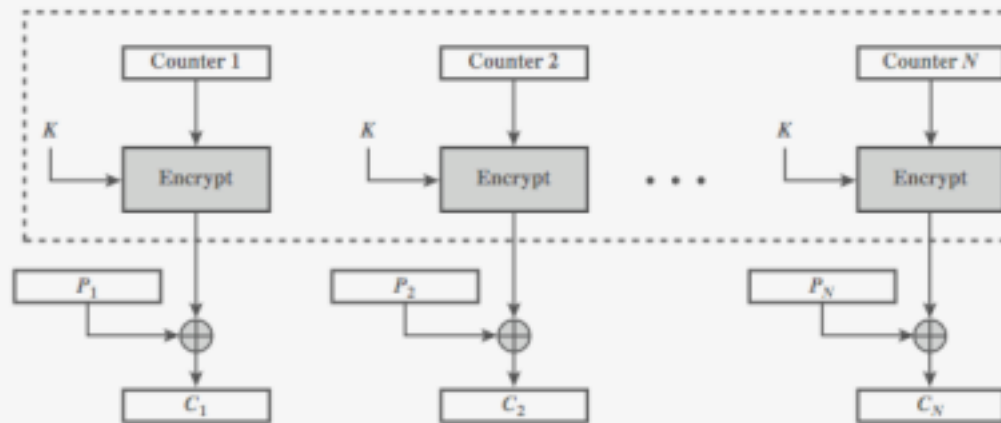
Advantages and Limitations of OFB

- bit errors do not propagate
- more vulnerable to message stream modification
- a variation of a Vernam cipher
 - hence must **never** reuse the same sequence (key+IV)
- sender & receiver must remain in sync
- originally specified with m-bit feedback
- subsequent research has shown that only **full block feedback** (ie OFB-64 or OFB-128) should ever be used

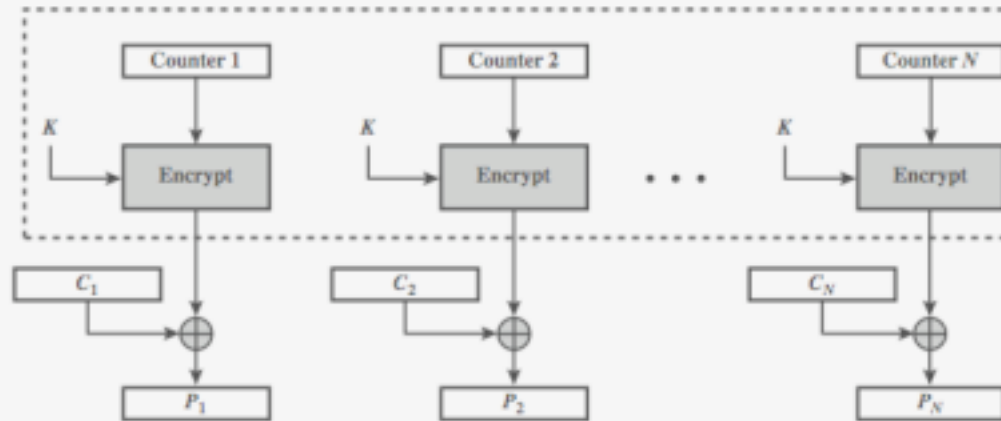
Counter (CTR)

- a “new” mode, though proposed early on
- similar to OFB but encrypts counter value rather than any feedback value
- must have a different key & counter value for every plaintext block (never reused)
$$C_i = P_i \text{ XOR } O_i$$
$$O_i = \text{DES}_{K1}(i)$$
- uses: high-speed network encryptions, ATM, IPSec, network security

Counter (CTR)



(a) Encryption



(b) Decryption



Advantages and Limitations of CTR

- **efficiency**
 - can do parallel encryptions in h/w or s/w
 - can preprocess in advance of need
 - good for bursty high speed links
- **random access to encrypted data blocks**
- **provable security (good as other modes)**
- **but must ensure never reuse key/counter values, otherwise could break (cf OFB)**

Summary

- **Cryptography.**
 - Classical
 - > transposition, substitution
 - Modern Private key
 - > DES
 - > AES
- **Attacks.**
 - Brute force
 - Cryptanalytic
- **Stream cipher**
- **Modes of Operation**
 - ECB, CBC, CFB, OFB, CTR

Further Reading

- **Chapter 20 of the textbook: *Computer Security: Principles and Practice*” by William Stallings & Lawrie Brown, Prentice Hall, 2015**
- **Acknowledgement: part of the materials presented in the slides was developed with the help of Instructor’s Manual and other resources made available by the author of the textbook.**