Cryptographic Algorithms

I. Symmetric Encryption Algorithms

Symmetric encryption uses a single Key for both encryption and decryption.

1) Substitution Cipher

a) Monoalphabetic Substitution Cipher

- Each character in the plaintext is replaced with a corresponding ciphertext character based on a predefined mapping

b) Increment Substitution Cipher (Caeser Cipher)

- plaintext : VOYAGER
- Key : +5 (each letter is shifted forward by 5 positions in the alphabet)
- Ciphertext : ATDFLJW
- Decryption rule : -5

c) Complex substitution Cipher using patterned Key

- plaintext : VOYAGER
- Key : "123" => +1, +2, +3, +1, +2, +3, +1
- Ciphertext : WQBBIHS

d) Simple polyalphabetic substitution

→ Vigenère Cipher

- Uses multiple caeser cipher
- plaintext : WINTER
- Key : KECA (repealed as needed)
- Key size : 4 => divide plaintext into blocks of 4 characters.
- M1 : WINT M2 : ER
- Shift each letter in the block by corresponding Key letter's value (K : +10, E : +4, C : +2, A : +0)
- Cipher text : GMPTOV

→ One - Time Pad

- Encrypts each character by combining it with a random key of the same length.
- Same principle of vigenère cipher
- Plaintext : HELLO
- Key : XMCKL
- Encryption

H + X = E
E + M = Q
L + C = N
L + K = V
O + L = Z

} Cipher Text: EQNVZ

2) Transposition Cipher

a) Simple Transposition

- Characters are rearranged based on Key's positional order
- Plain Text : SECURITY
- Key CBVA (Size = 4)
- Alphabetical order : 3 2 4 1
- Divide plain text : SECU RITY
- Rearrange (3 → 2 → 4 → 1):
- ↳ CipherText : CEUSTIYR

b) Columnar Transposition (Important)

- Plaintext written in rows, then columns are reorder based on the Key
- Plaintext : HELLO WORLD
- Key : KEY (Size : 3)
- alphabetical order : E K Y (1 2 3) => 2 1 3

- Encryption
  - → write plaintext as rows =>
  - → Read the ciphertext by column in order (1, 2, 3)
  - => Cipher Text :

EORX HLOD LWLX

2	1	3
H	E	L
L	O	W
O	R	L
D	X	X

- Decryption:
  - → Write the ciphertext as columns by order ↑
  - → Nb. of rows = len(ciphertext)/len(Key) (12/3 = 4)
  - → Read the plaintext by row: HELLO WORLD

2	1	3
H	E	L
L	O	W
O	R	L
D	X	X

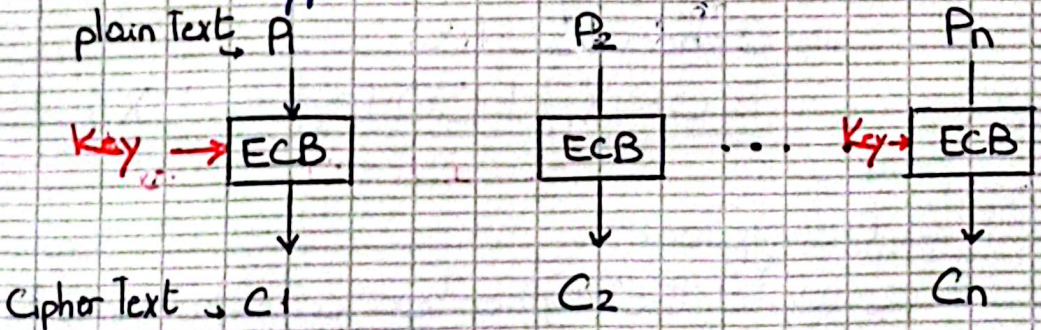
3) Data Encryption Standard (DES)

- Key size : 56 bit key
- Block size : 64-bit blocks
- Process:
  - → Uses Feistel structure with 16 rounds of encryp.
- Vulnerabilities: Brute-force attacks due to small key size
- Make DES more secure:
  - → use three keys sequentially (3-DES)
  - ↳ 3DES: Key size 112 or 168-bit
  - → use cipher-block chaining

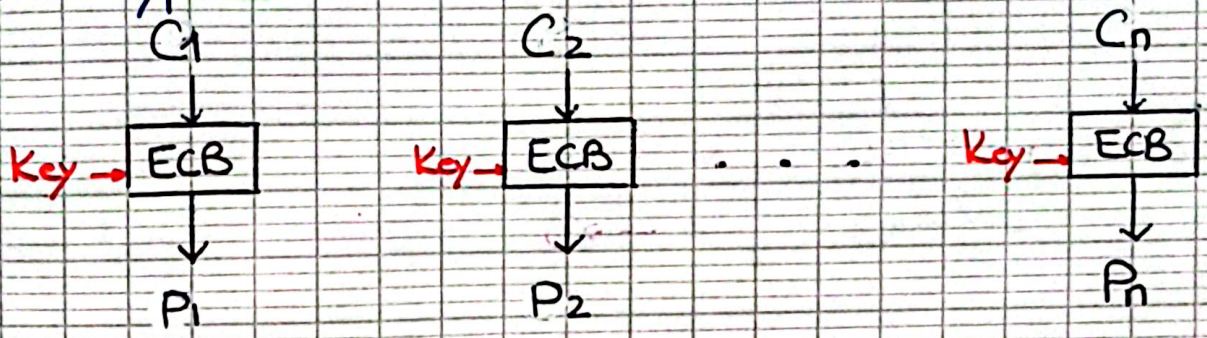
4) Block Cipher Modes of Operation (Important)

a) ECB (Electronic CodeBook)

- Process:
  - → Message broken into independent blocks
  - → Each block is encrypted independently using the same key
- Formula: $C_i = K(P_i)$
  - C_i ← Cipher Text
  - K ← Key
  - P_i ← plain text
- Encryption:



- Decryption



- Example:
  - Plain Text : 1001 1101
  - Key : swap the first two bits with the last two bits

$C_i = K(P_i)$

$C_1 = K(P_1)$ $\qquad$ $C_2 = K(P_2)$

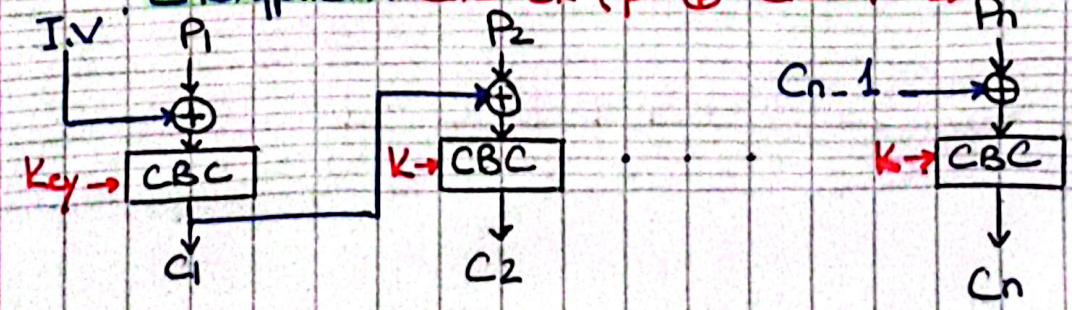
$C_1 = 0110$ $\qquad$ $C_2 = 0111$

$\Rightarrow$ Cipher Text : 0110 0111

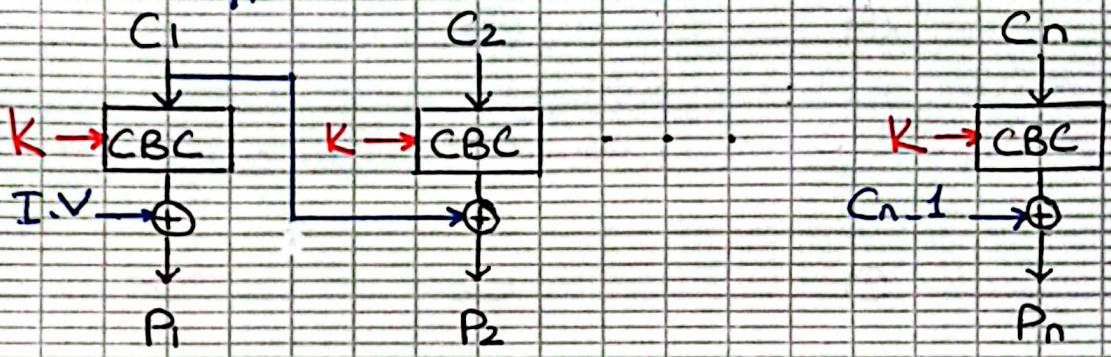
- Usecases: Transmission of single values.

b) Cipher Block Chaining (CBC)

- Process:
  - Message is broken into blocks based on the size of the Initial Value
  - Each plain text block is XORed with the previous cipher text block before encryption
- Encryption: $C_i = E_K(p_i \oplus C_{i-1}) \Rightarrow C_0 = I.V$



.Decryption: $P_i = D_K(C_i) \oplus C_{i-1}$ $\Rightarrow C_0 = I.V$



.Example

→ I.V = 0011 (Size = 4)

→ P = 1101 / 1010

→ K : Reverse the order of each block

⊕ ⇓
11 : 0
00 : 0
10 : 1
01 : 1

→ Encryption

$C_1 = E_K(P_1 \oplus I.V)$

$C_1 = E_K\left(\begin{matrix}1&1&0&1\\ \oplus\,0&0&1&1\end{matrix}\right)$

$C_1 = E_K(1110)$

$C_1 = 0111$

$C_2 = E_K(P_2 \oplus C_1)$

$C_2 = E_K\left(\begin{matrix}1&0&1&0\\ \oplus\,0&1&1&1\end{matrix}\right)$

$C_2 = E_K(1101)$

$C_2 = 1011$

$\Rightarrow C = 01111011$

→ Decryption:

$P_1 = D_K(C_1) \oplus IV$

$= 1\ 1\ 1\ 0$

$\oplus\ 0\ 0\ 1\ 1$

$= 1\ 1\ 0\ 1$

$P_2 = D_K(C_2) \oplus C_1$

$= 1\ 1\ 0\ 1$

$\oplus\ 0\ 1\ 1\ 1$

$= 1\ 0\ 1\ 0$

$\Rightarrow P = 1\ 1\ 0\ 1\ 1\ 0\ 1\ 0$

- Use cases: Bulk data encryption, authentication

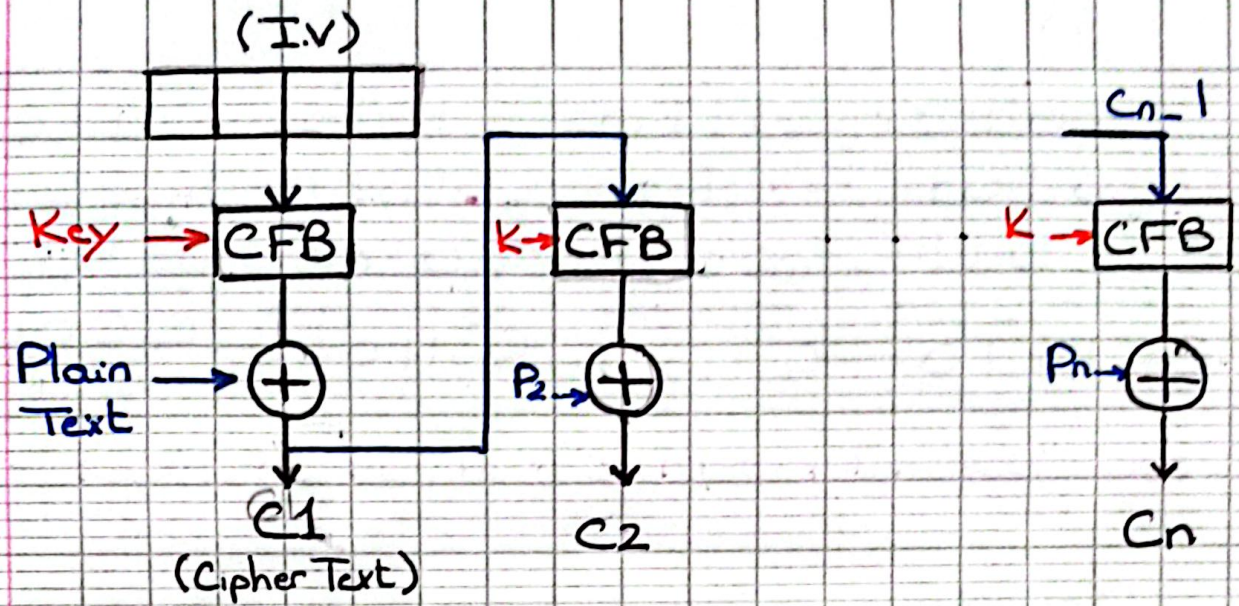
c) Cipher Feed Back (CFB)

- Process
  - Message is treated as a stream of bits (small chuncks of data) instead of entire blocks.
  - We start with an Initialization Vector (IV)
  - Encrypt the IV using the block cipher
  - The encrypted IV is XORed with the first chunk of plaintext to create the first piece of ciphertext
  - The ciphertext produced replaces the IV
  - This new value is encrypted again and XORed withe next plaintext chunk
  - This process continues for the enlire message

- Encryption:

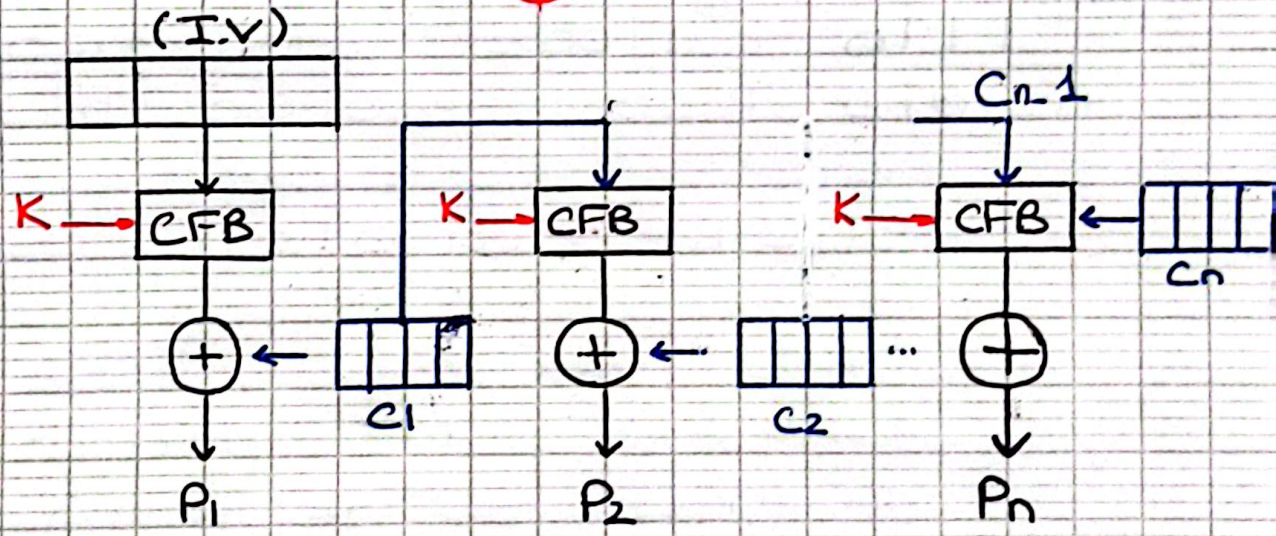
$$C_i = E_K(c_{i-1}) \oplus P_i$$

$$C_0 = I.V$$



Decryption:

$P_i = E_K(C_{i-1}) \oplus C_i \qquad C_0 = I.V$



Example:

I.V = 1010

M = 1011/0001/0100/101

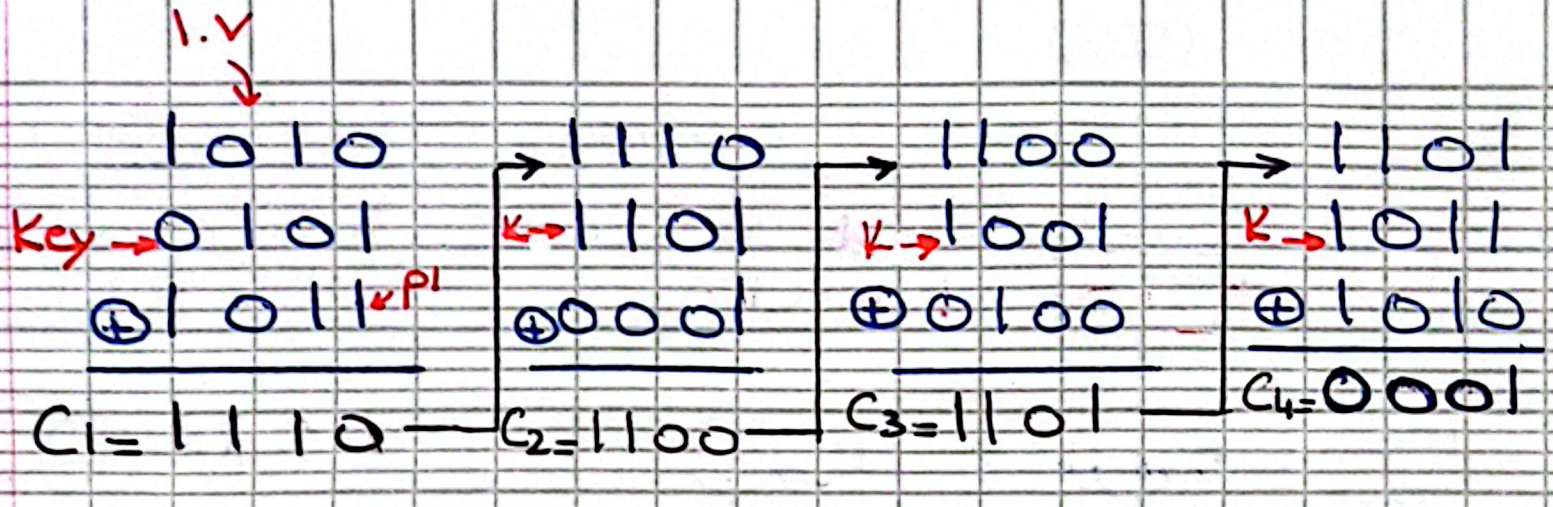
Key: 1 → 2

2 → 3

3 → 4

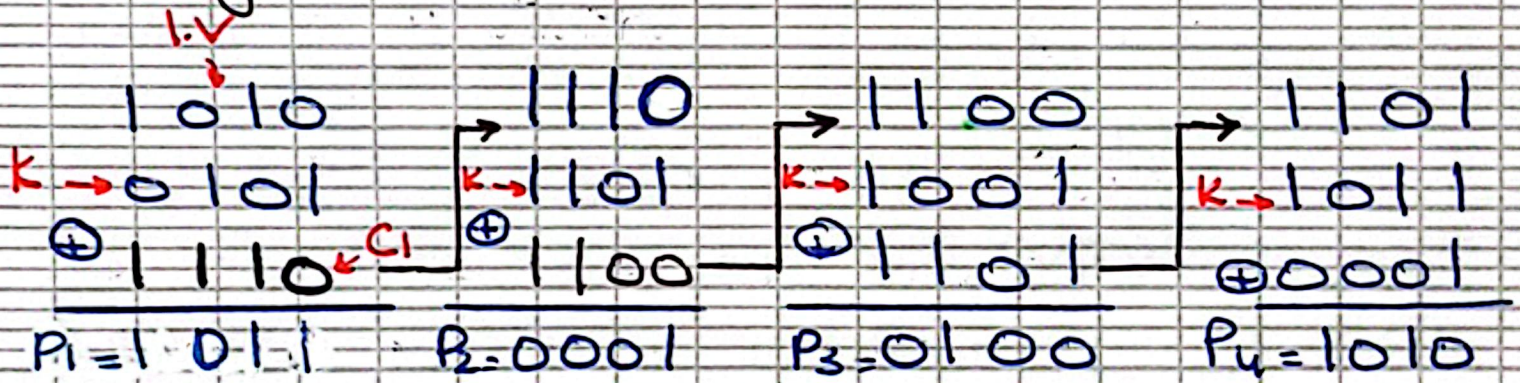
4 → 1

· Encryption :



<=> C = 1110 1100 1101 0001 *

· Decryption :



<=> P = 1011 0001 0100 1010

Difference Between CBC and CFB

Feature	CFB	CBC
Encryption Style	Works like a stream cipher (encrypts small chuncks of data)	Works like a block cipher (encrypts full block of data)
Input to Block cipher	Previous ciphertext is encrypted to generate a keyStream	Previous cipher text is XORed with the next plain text before encryption
Decryption Process	Same encryption process, XOR with ciphert. to get plaint	Decrypts ciphert. then XOR with the previous one to get plain text
Padding	Not required (can work with different data lengths)	Required (plain text must be padded to fit block size)
Speed	Faster for streaming encry.	More efficient for bulk data encry.