# Block cipher Modes of operation

# Objectives

❑ To discuss five modes of operation designed to be used with modern block ciphers.

❑ To define which mode of operation creates stream ciphers out of the underlying block ciphers.

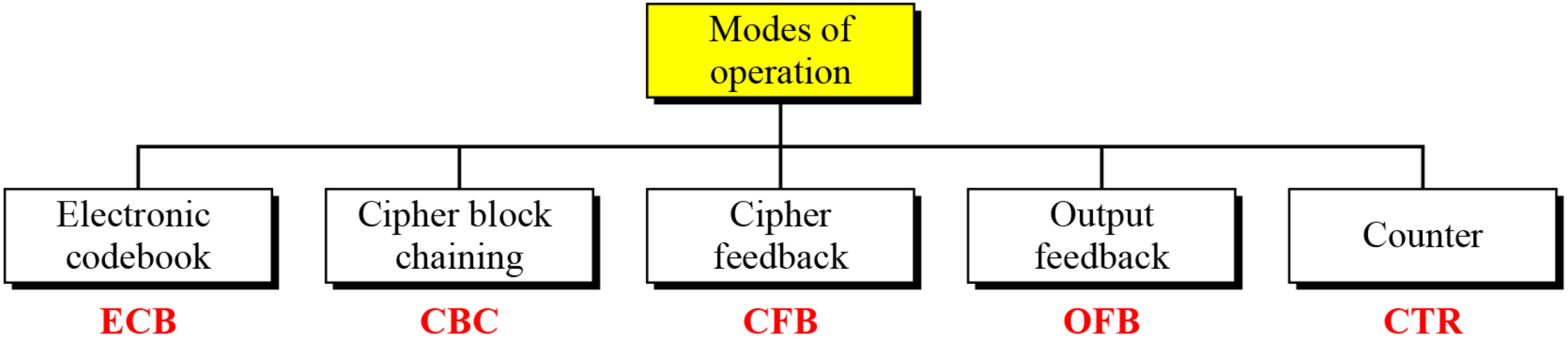
❑ To discuss the security issues and the error propagation of different modes of operation.

# USE OF MODERN BLOCK CIPHERS

- Symmetric-key encipherment can be done using modern block ciphers.
- Modes of operation have been devised to encipher text of any size employing either DES or AES.

# Modes of Operation

**Figure** ***Modes of operation***

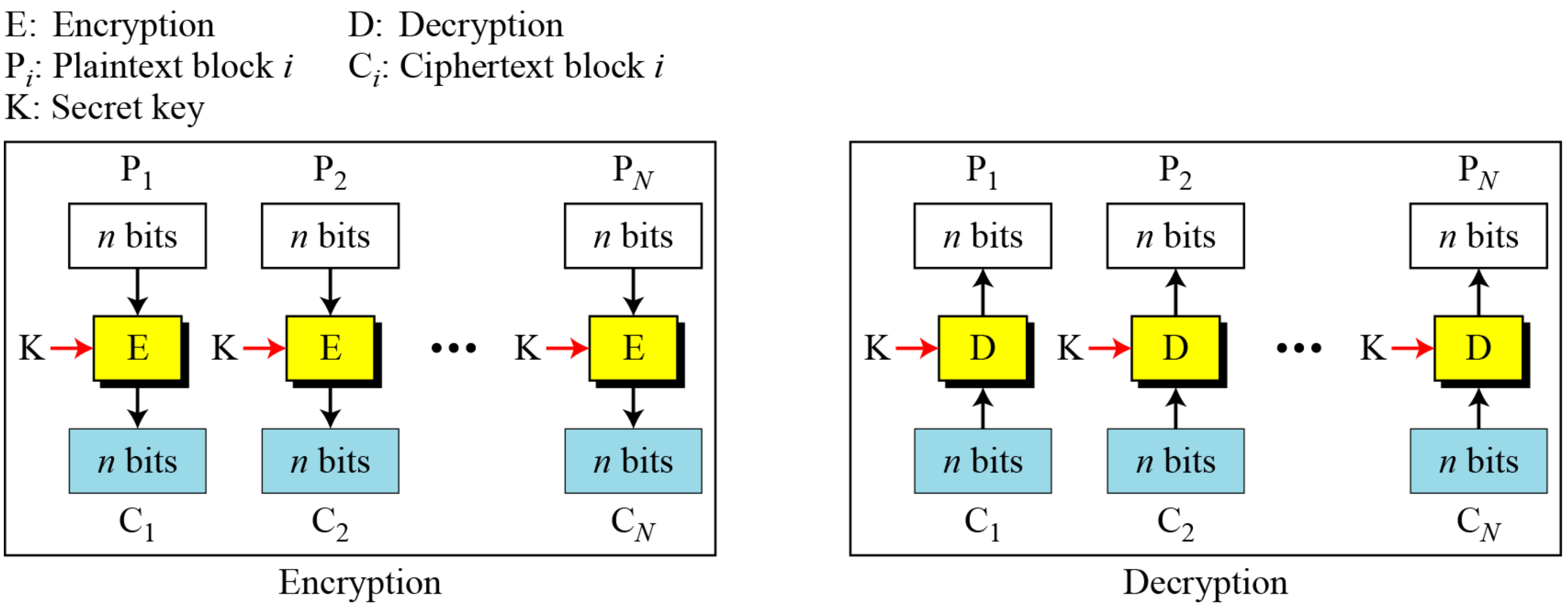


# I. Electronic Codebook (ECB) Mode

The simplest mode of operation is called the electronic codebook (ECB) mode.

Encryption: $C_i = E_K(P_i)$ Decryption: $P_i = D_K(C_i)$

**Figure** ***Electronic codebook (ECB) mode***

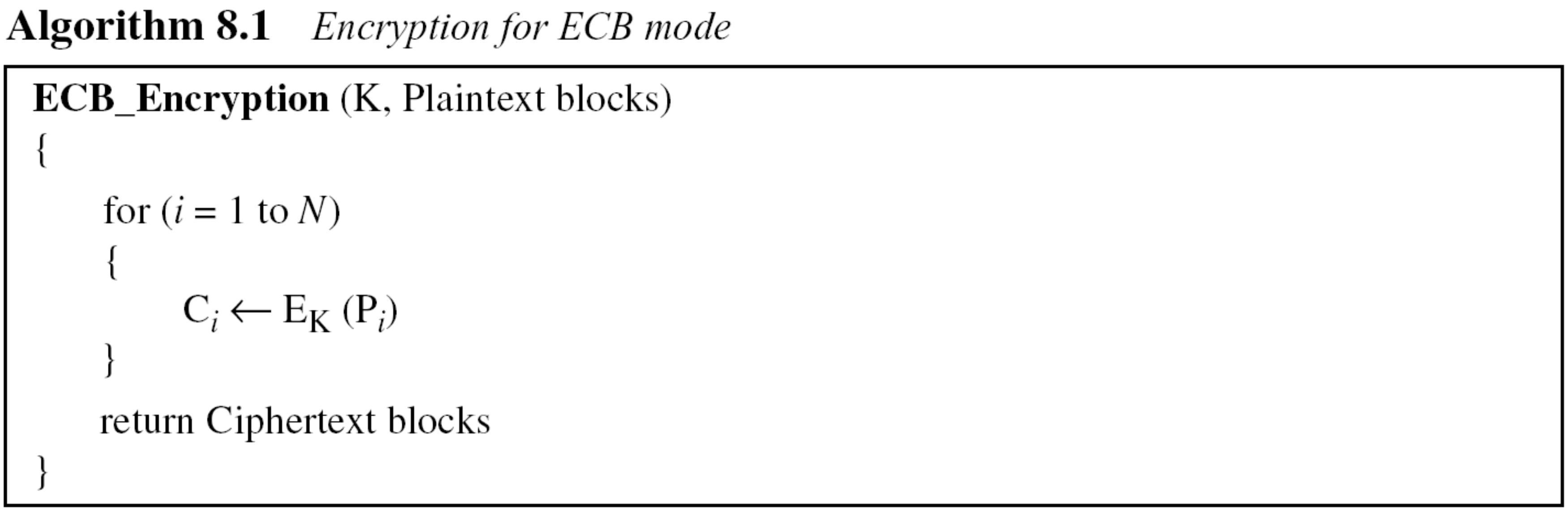


# *Security issues in ECB mode*

1. Patterns at block level are preserved.
2. Block independency creates opportunities for attacker to exchange some cipher text blocks without knowing the key.

# Error Propagation

A single bit error in transmission can create errors in several (normally half or all bits) in the corresponding block. However, the error does not have any effect on the other blocks.



**Algorithm 8.1** *Encryption for ECB mode*

```
ECB_Encryption (K, Plaintext blocks)
{
        for ($i$ = 1 to $N$)
        {
                $C_i \leftarrow E_K (P_i)$
        }
        return Ciphertext blocks
}
```

# *Applications*

1. If message is short enough to fit in one block, security issues and error propagation are tolerable.
2. It is useful where records need to be encrypted before they are stored in a database.
3. ECB allows parallel processing, if we want to create a very huge encrypted database.

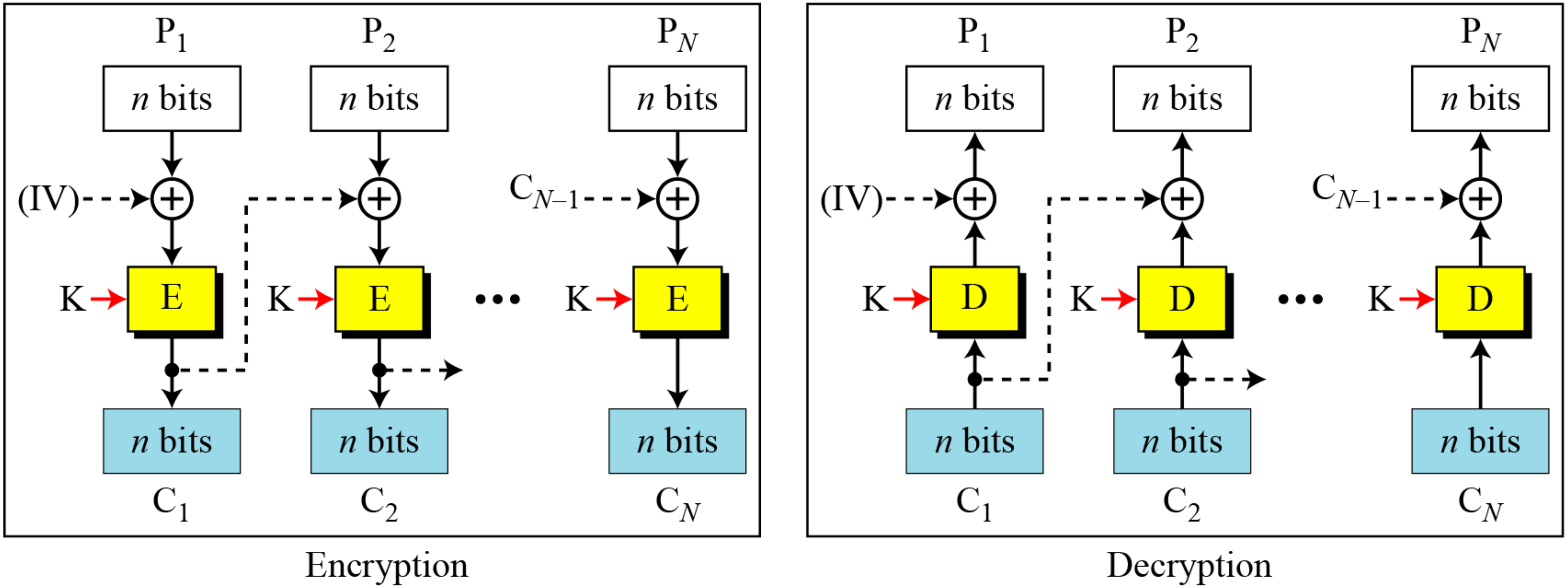
# II. Cipher Block Chaining (CBC) Mode

*In CBC mode, each plaintext block is exclusive-ored with the previous ciphertext block before being encrypted.*

**Figure** ***Cipher block chaining (CBC) mode***

E: Encryption  
$P_i$: Plaintext block *i*  
K: Secret key

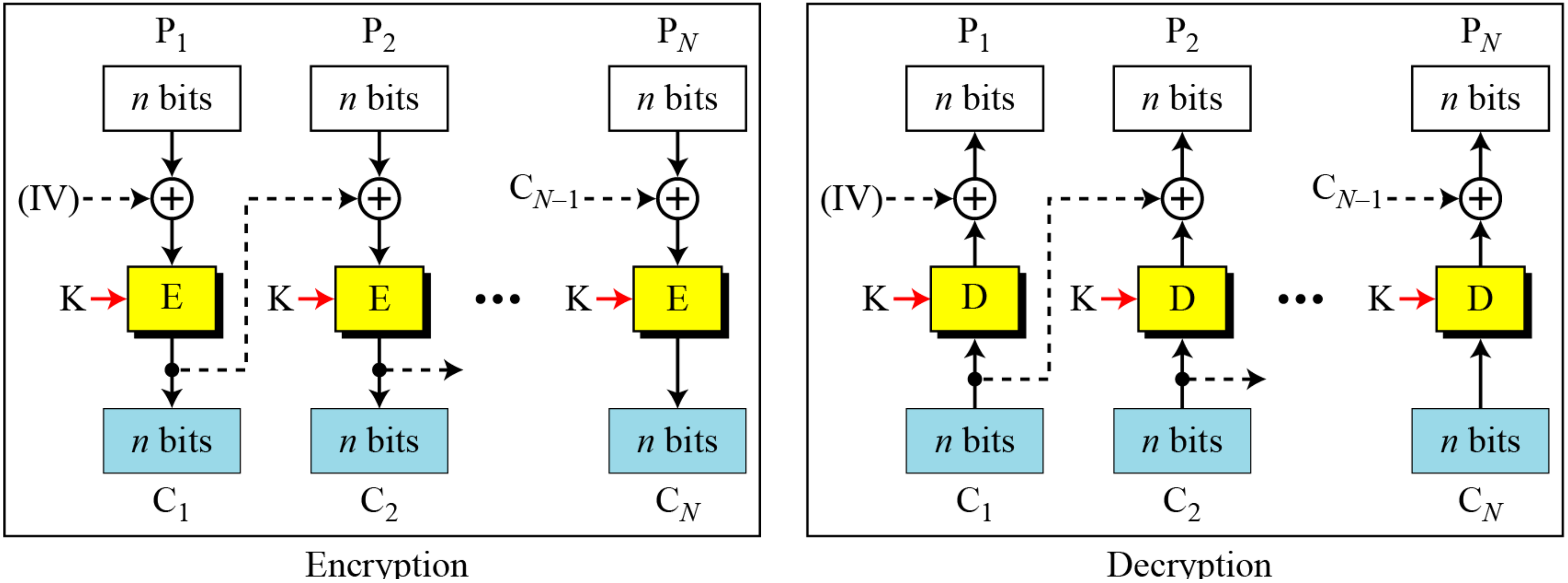
D : Decryption  
$C_i$ : Ciphertext block *i*  
IV: Initial vector ($C_0$)



# Continued

**Figure** ***Cipher block chaining (CBC) mode***

E: Encryption  
D : Decryption  
$P_i$: Plaintext block *i*  
$C_i$ : Ciphertext block *i*  
K: Secret key  
IV: Initial vector ($C_0$)



**Encryption:**  
$C_0 = \text{IV}$  
$C_i = E_K(P_i \oplus C_{i-1})$

**Decryption:**  
$C_0 = \text{IV}$  
$P_i = D_K(C_i) \oplus C_{i-1}$

## *Initialization Vector (IV)*

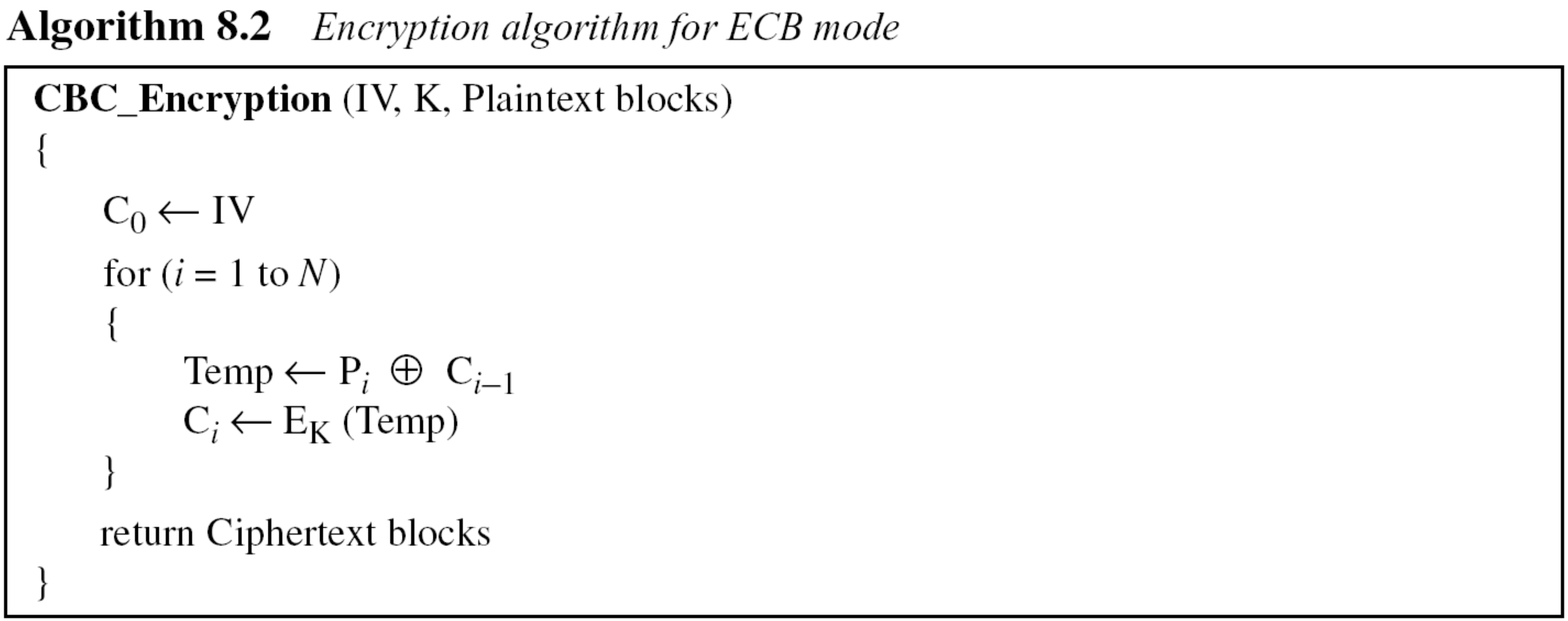
***The initialization vector (IV) should be known by the sender and the receiver.***

# *Security Issues*

1. Patterns at block levels are not preserved. However if two messages are equal, their encipherment is the same if they use same IV.
2. Attacker can add some cipher text blocks to the end of cipher text stream.

# Error Propagation

In CBC mode, a single bit error in ciphertext block $C_j$ during transmission may create error in most bits in plaintext block $P_j$ during decryption.



**Algorithm 8.2** *Encryption algorithm for ECB mode*

```
CBC_Encryption (IV, K, Plaintext blocks)
{
    $C_0 \leftarrow IV$
    for ($i$ = 1 to $N$)
    {
        $Temp \leftarrow P_i \oplus C_{i-1}$
        $C_i \leftarrow E_K(Temp)$
    }
    return Ciphertext blocks
}
```

# Applications

1. It is not used when we need parallel processing because it uses chaining mechanism.
2. It is not used to encrypt and decrypt random-access files records because encipherment here require access to previous records.
3. It is used for authentication purpose.

# *III. Cipher Feedback (CFB) Mode*

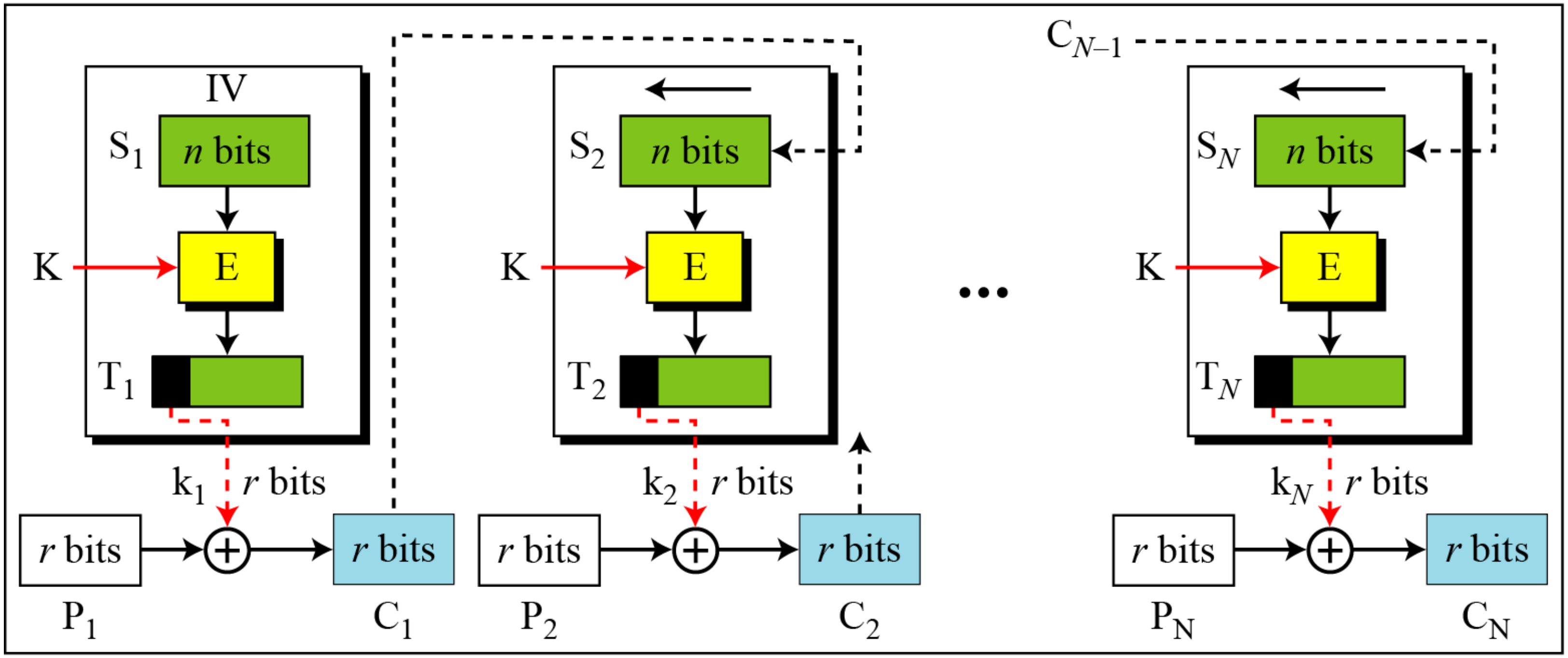
In some situations, we need to use DES or AES as secure ciphers, but the plaintext or ciphertext block sizes are to be smaller.

**Figure** ***Encryption in cipher feedback (CFB) mode***

E : Encryption
$P_i$: Plaintext block *i*
K: Secret key

D : Decryption
$C_i$ : Ciphertext block *i*
IV: Initial vector ($S_1$)

$S_i$: Shift register
$T_i$: Temporary register



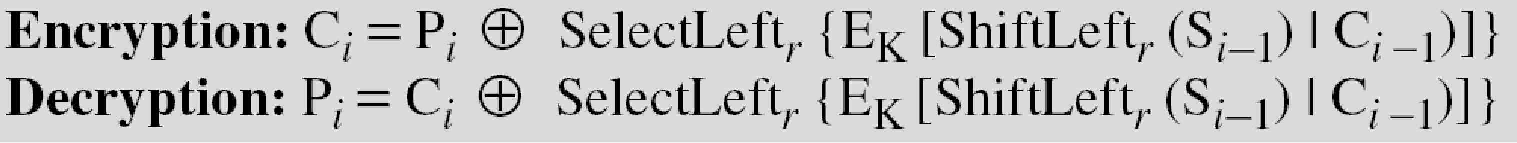
Encryption

# *Continued*

***Note***

**In CFB mode, encipherment and decipherment use the encryption function of the underlying block cipher.**

***The relation between plaintext and ciphertext blocks is shown below:***



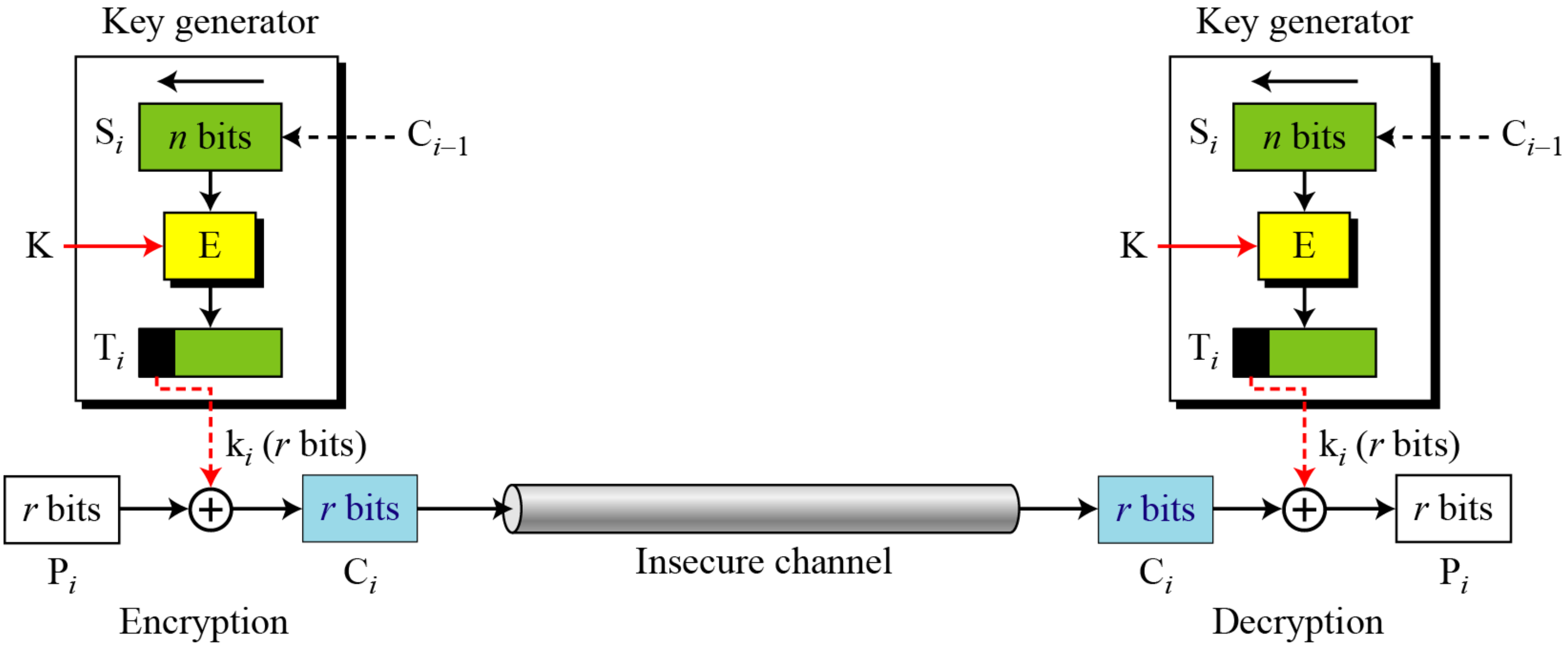
**Encryption:** $C_i = P_i \oplus \text{SelectLeft}_r\,\{E_K\,[\text{ShiftLeft}_r\,(S_{i-1}) \mid C_{i-1})]\}$

**Decryption:** $P_i = C_i \oplus \text{SelectLeft}_r\,\{E_K\,[\text{ShiftLeft}_r\,(S_{i-1}) \mid C_{i-1})]\}$

# *Continued*

## *CFB as a Stream Cipher*

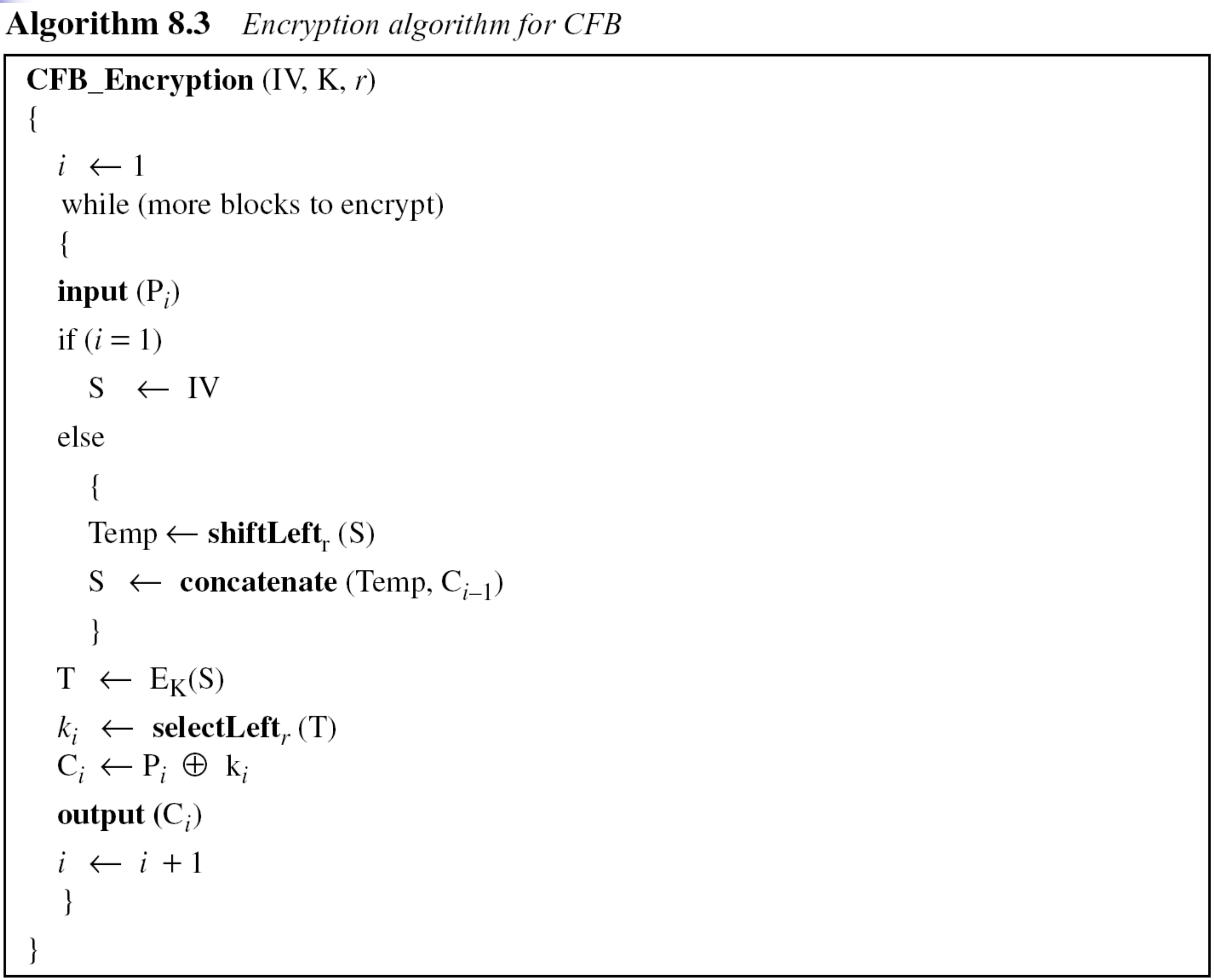
**Figure** ***Cipher feedback (CFB) mode as a stream cipher***



# Security Issues

1. Patterns at block level are not preserved.
2. Need to use different IV each time we send a message.
3. Attacker can add some cipher text block to the end of the cipher text stream.

# Continued



**Algorithm 8.3** *Encryption algorithm for CFB*

```
CFB_Encryption (IV, K, $r$)
{
   $i \leftarrow 1$
   while (more blocks to encrypt)
   {
   input ($P_i$)
   if ($i = 1$)
      S ← IV
   else
      {
      Temp ← shiftLeft$_r$ (S)
      S ← concatenate (Temp, $C_{i-1}$)
      }
   T ← $E_K$(S)
   $k_i \leftarrow$ selectLeft$_r$ (T)
   $C_i \leftarrow P_i \oplus k_i$
   output ($C_i$)
   $i \leftarrow i + 1$
   }
}
```

# *Error Propagation*

1. A single bit error in cipher text block during transmission creates a single bit error (at same position) in the plaintext block.

# *Application*

1. Can be used to encipher blocks of small size such as one character or bit at a time. No need of padding because size of plaintext block is normally fixed(8 for character or 1 for bit)

# IV. Output Feedback (OFB) Mode

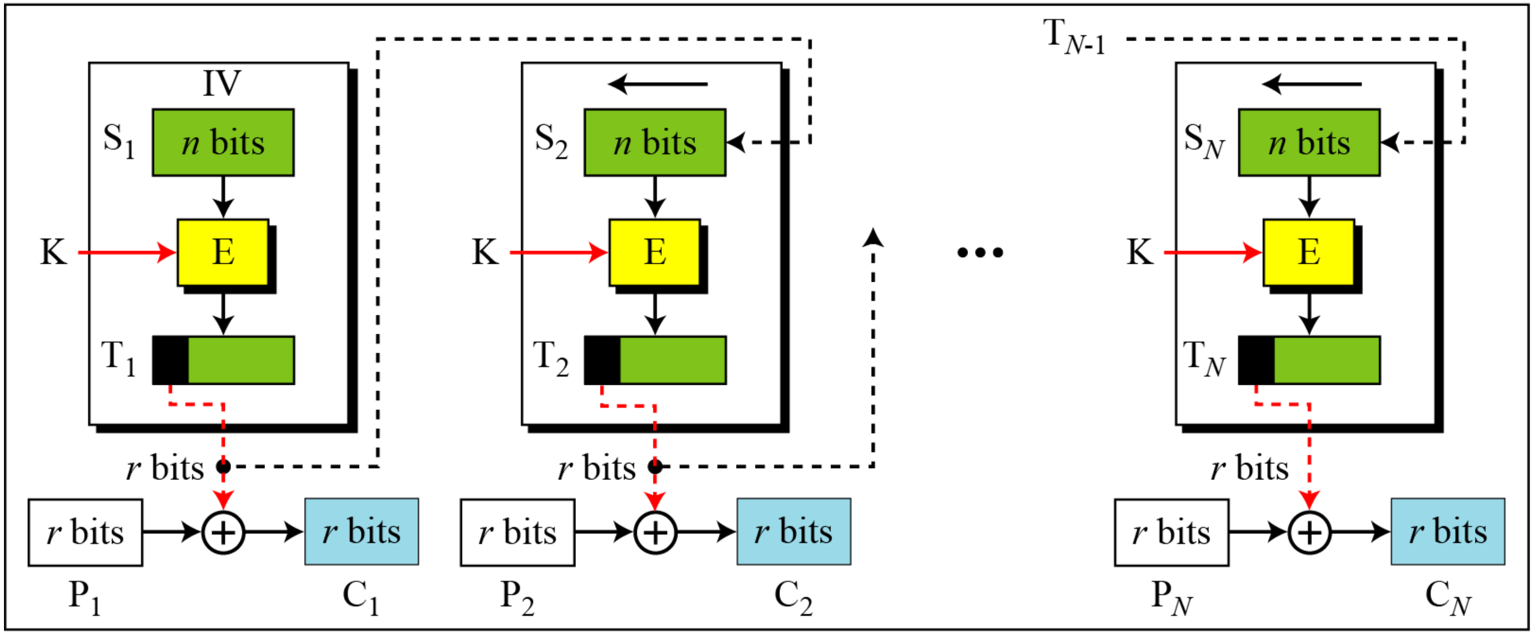
In this mode each bit in the ciphertext is independent of the previous bit or bits. This avoids error propagation.

**Figure** ***Encryption in output feedback (OFB) mode***

E : Encryption
$P_i$: Plaintext block i
K: Secret key

D : Decryption
$C_i$: Ciphertext block i
IV: Initial vector ($S_1$)

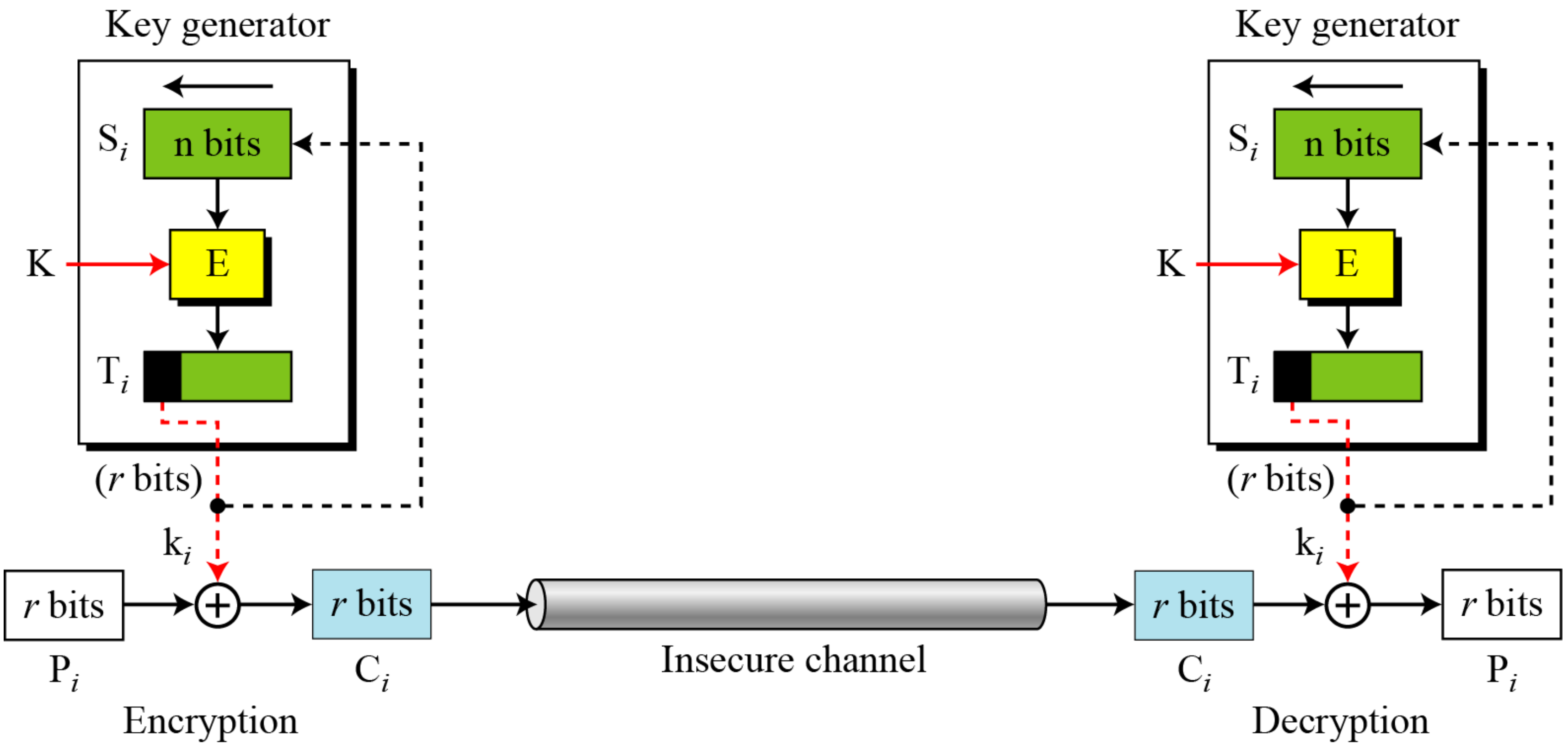
$S_i$: Shift register
$T_i$: Temporary register



Encryption

# OFB as a Stream Cipher

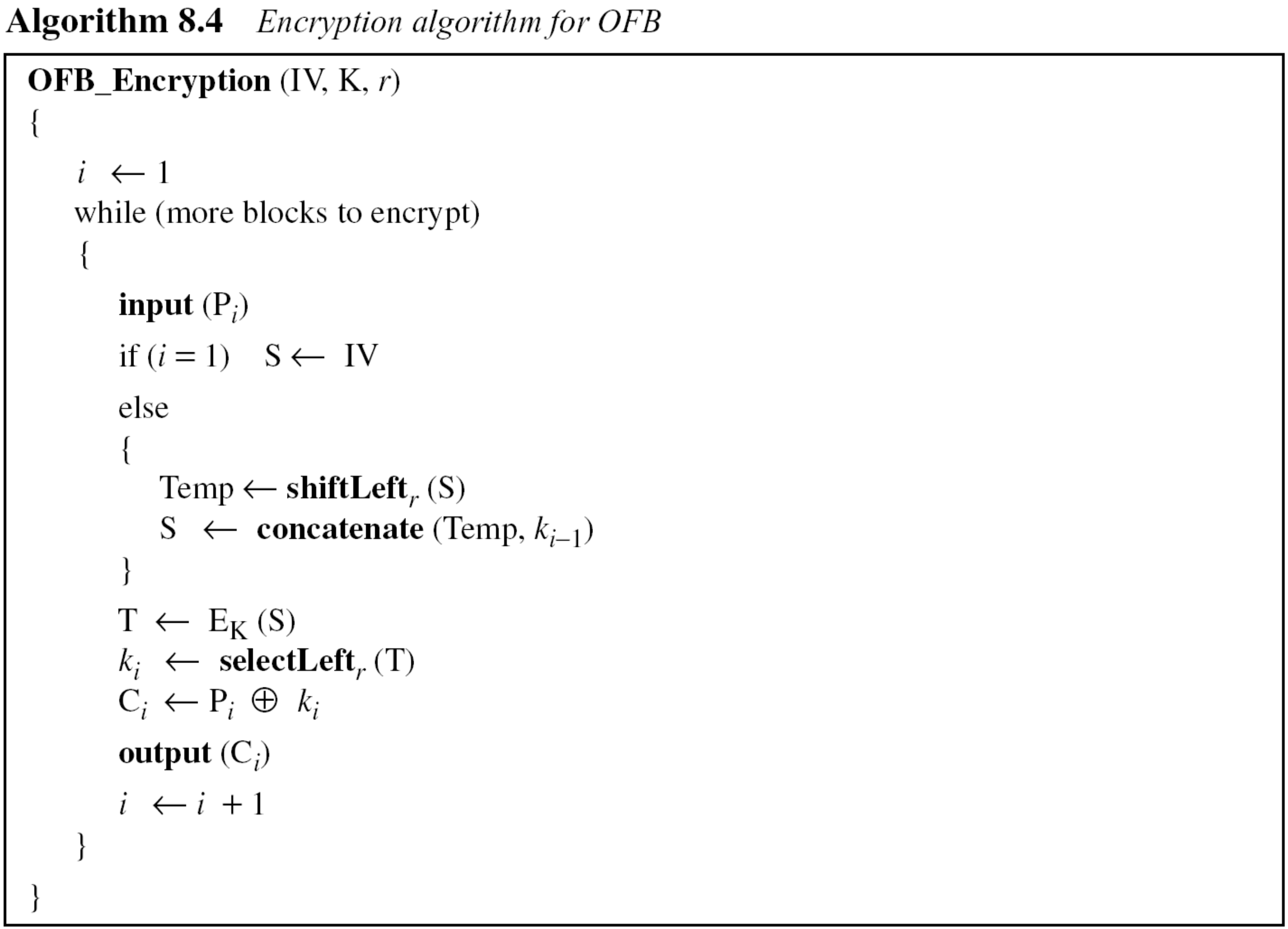
**Figure** ***Output feedback (OFB) mode as a stream cipher***



# Security issues

1. Like CFB, patterns at block level are not preserved.
2. Any change in cipher text affects the plaintext encrypted at the receiver side.

# Continued



**Algorithm 8.4** *Encryption algorithm for OFB*

```
OFB_Encryption (IV, K, $r$)
{
    $i \leftarrow 1$
    while (more blocks to encrypt)
    {
        input ($P_i$)
        if ($i = 1$)    $S \leftarrow IV$
        else
        {
            $Temp \leftarrow \textbf{shiftLeft}_r (S)$
            $S \leftarrow \textbf{concatenate} (Temp, k_{i-1})$
        }
        $T \leftarrow E_K (S)$
        $k_i \leftarrow \textbf{selectLeft}_r (T)$
        $C_i \leftarrow P_i \oplus k_i$
        output ($C_i$)
        $i \leftarrow i + 1$
    }
}
```

# *Error Propagation*

1. A single error in the cipher text affects only the corresponding bit in the plaintext.

# Application

1.Can be used to encipher blocks of small size such as one character or bit at a time. No need of padding because size of plaintext block is normally fixed(8 for character or 1 for bit)

# V. Counter (CTR) Mode

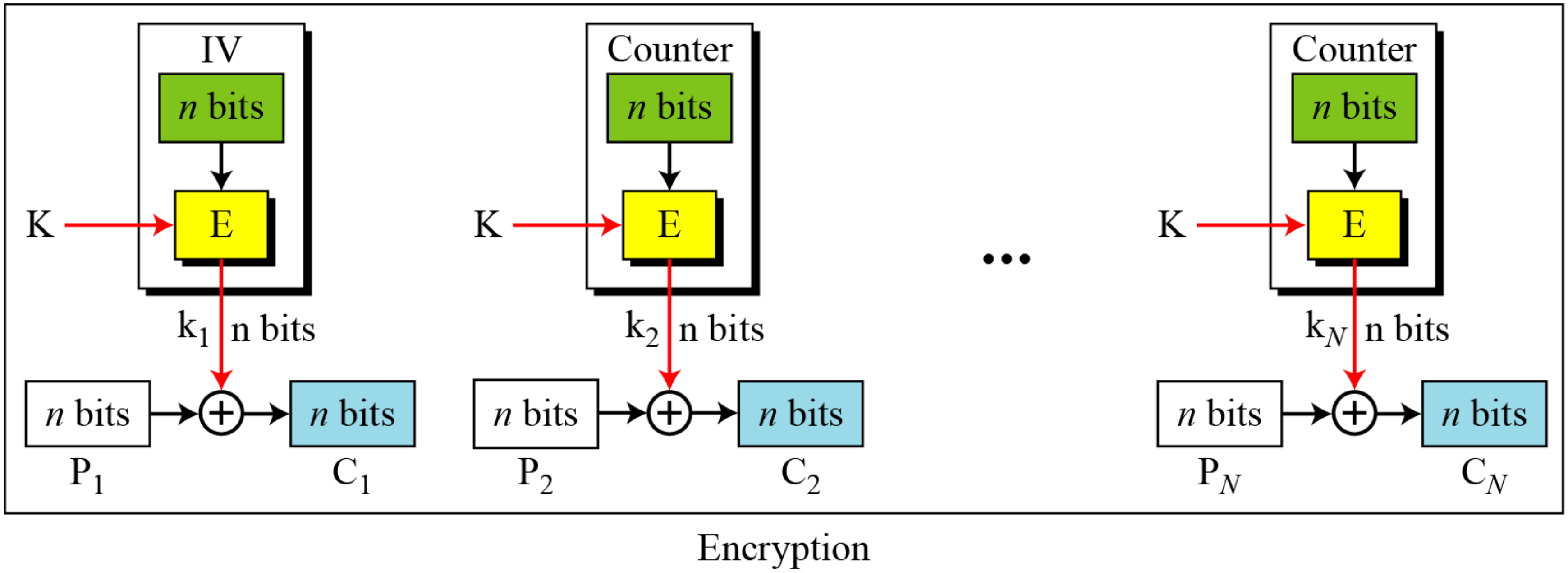
In the counter (CTR) mode, there is no feedback. The pseudorandomness in the key stream is achieved using a counter.

**Figure** ***Encryption in counter (CTR) mode***

E : Encryption
$P_i$ : Plaintext block *i*
K : Secret key

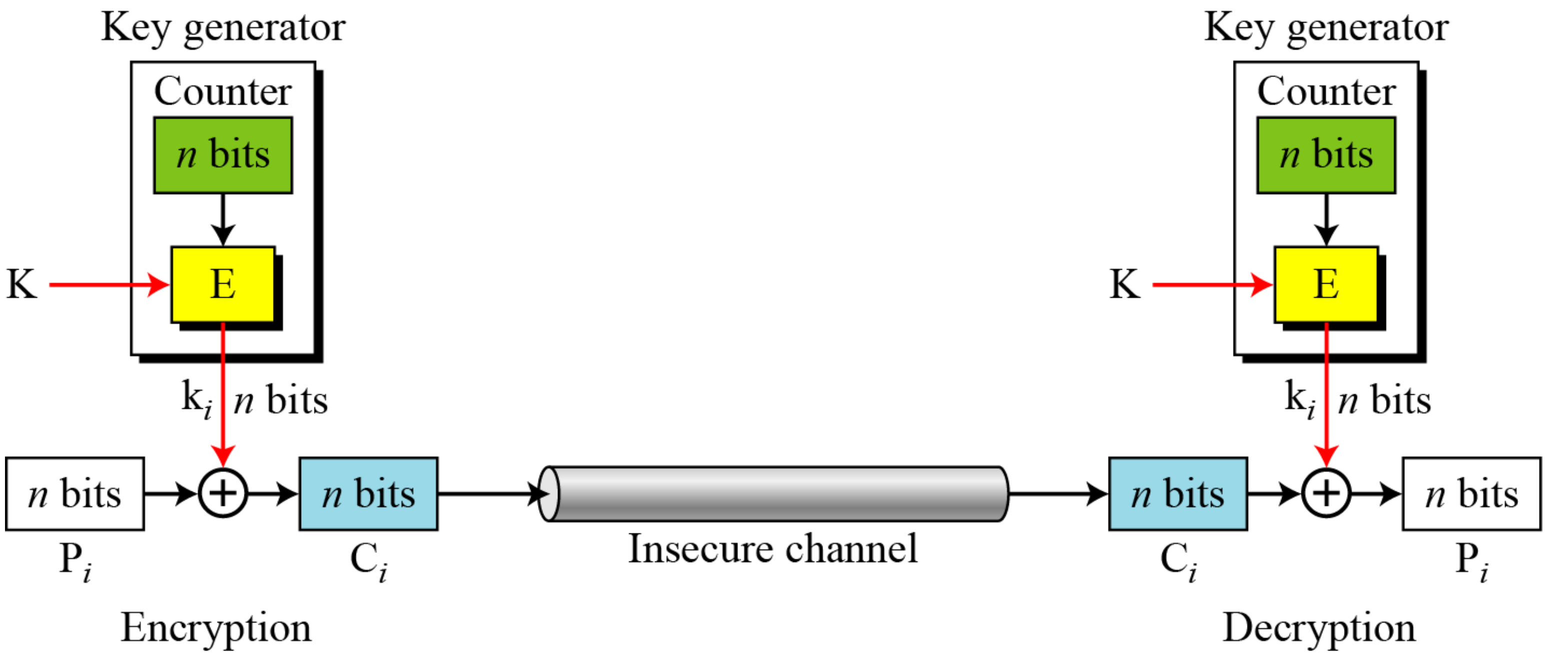
IV: Initialization vector
$C_i$ : Ciphertext block *i*
$k_i$ : Encryption key *i*

The counter is incremented for each block.



# Continued

**Figure** ***Counter (CTR) mode as a stream cipher***



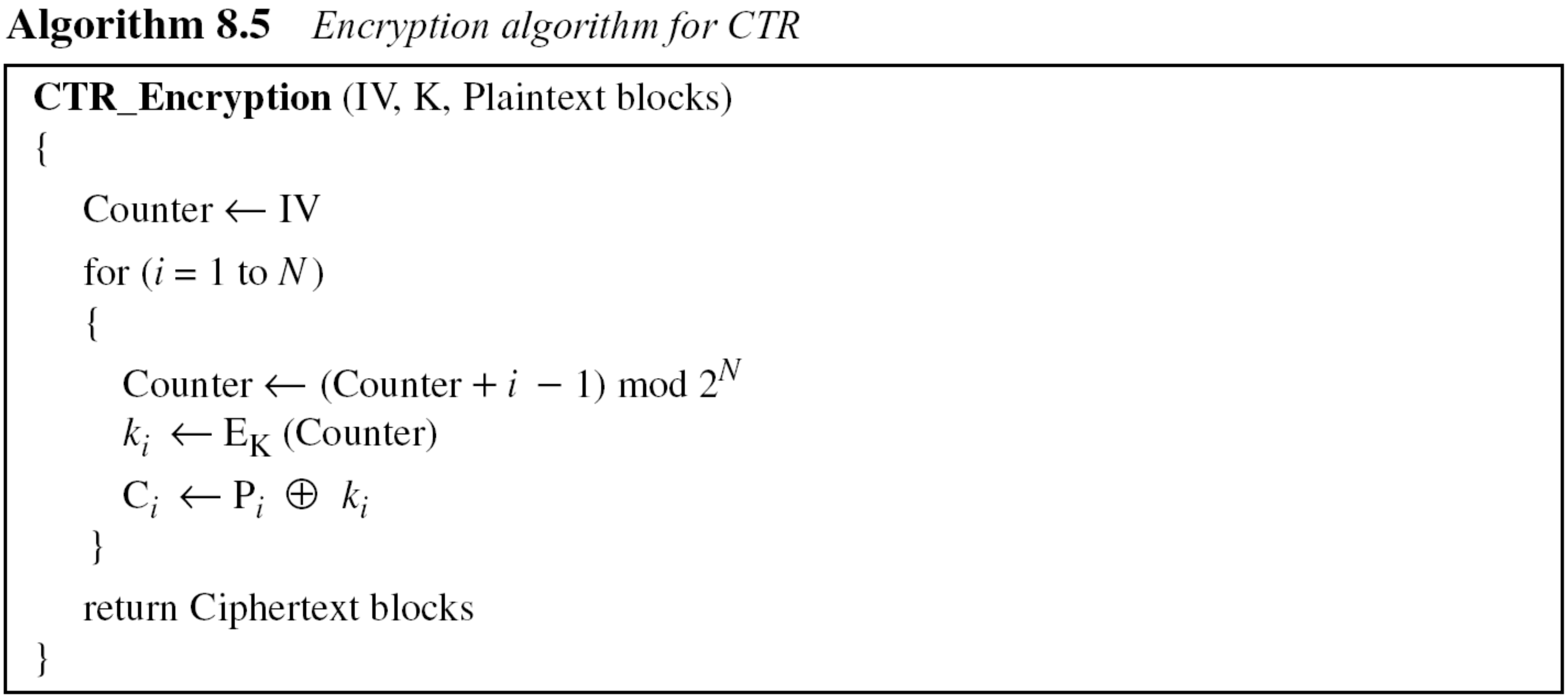
# *Security issues*

1. Like CFB, patterns at block level are not preserved.
2. Any change in cipher text affects the plaintext encrypted at the receiver side.

# *Error Propagation*

Single error in cipher text affects only the corresponding bit in the plaintext.

# Continued



**Algorithm 8.5** *Encryption algorithm for CTR*

```
CTR_Encryption (IV, K, Plaintext blocks)
{
    Counter ← IV
    for ($i$ = 1 to $N$)
    {
        Counter ← (Counter + $i$ − 1) mod $2^N$
        $k_i$ ← $E_K$ (Counter)
        $C_i$ ← $P_i \oplus k_i$
    }
    return Ciphertext blocks
}
```