Chapter 8

Encipherment Using Modern Symmetric-Key Ciphers

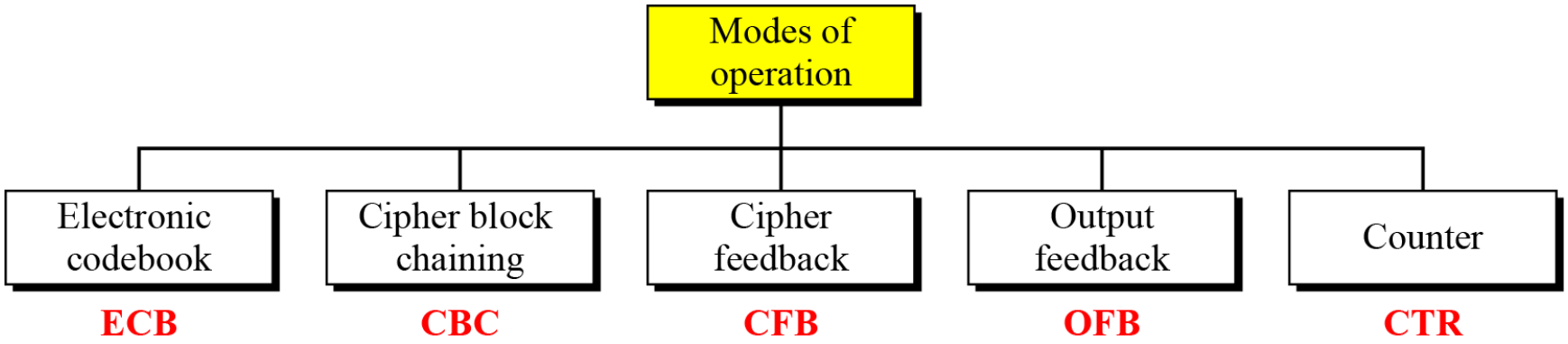
Chapter 8 Objectives

- Block length is fixed (*n*-bit)
- How to encrypt large messages?
  - Partition into *n*-bit blocks
  - Choose mode of operation
    - Electronic Codebook (ECB),
    - Cipher-Block Chaining (CBC),
    - Cipher Feedback (CFB),
    - Output Feedback (OFB),
    - Counter (CTR)

  -- Modes of operation have been devised to encipher text of any size employing either DES or AES.
- Two stream ciphers used for real-time processing of data.

8-1 Continued

Figure 8.1 *Modes of operation*



Evaluation criteria

- Identical messages
  - under which conditions ciphertext of two identical messages are the same
- Chaining dependencies
  - how adjacent plaintext blocks affect encryption of a plaintext block
- Error propagation
  - resistance to channel noise
- Efficiency
  - preprocessing
  - parallelization: random access

Electronic Codebook Book (ECB)

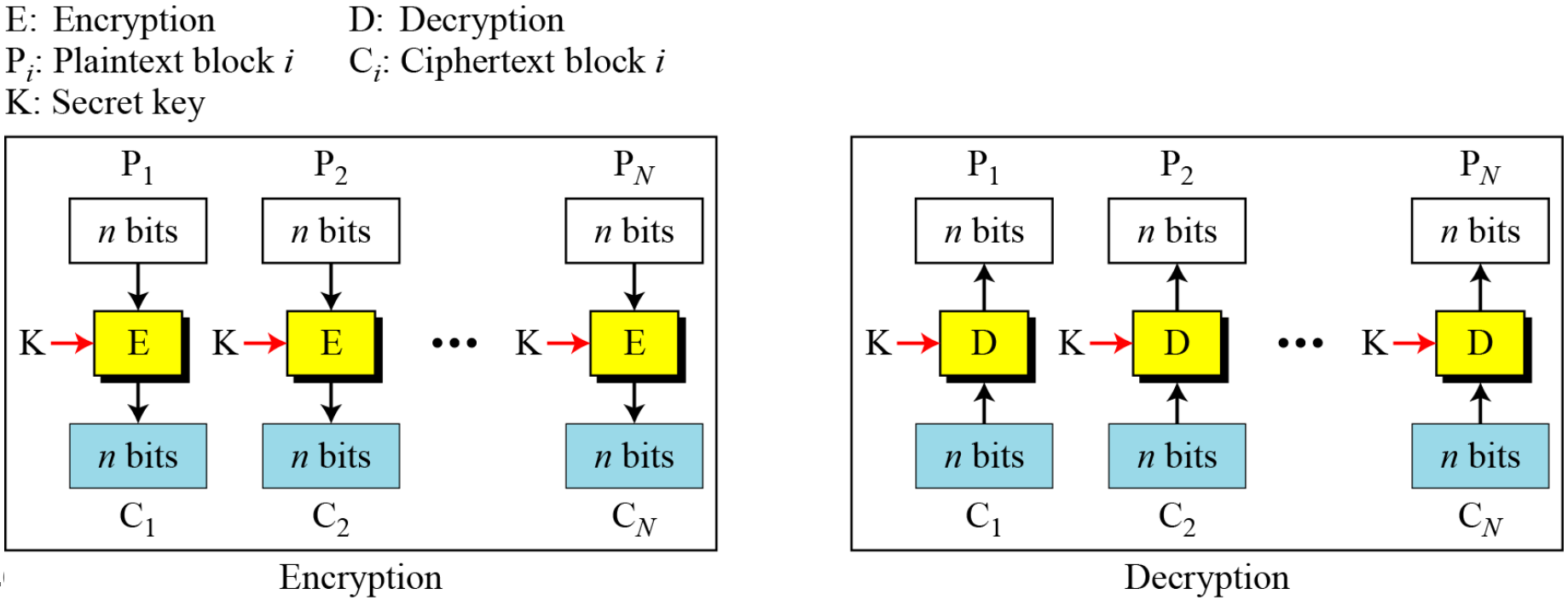
- message is broken into independent blocks which are encrypted
- each block is a value which is substituted, like a codebook, hence name
- each block is encoded independently of the other blocks using the same key
- uses: secure transmission of single values

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 8.2 *Electronic codebook (ECB) mode*



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

Electronic Codebook (ECB)

- Does not hide data patterns, unsuitable for long messages
  - Wiki example: pixel map using ECB

Plain text

ECB mode

Other modes

- Susceptible to replay attacks
  - Example: a wired transfer transaction can be replayed by resending the original message)

Electronic Codebook (ECB)

Example

Assume that Eve works in a company a few hours per month (her monthly payment is very low). She knows that the company uses several blocks of information for each employee in which the seventh block is the amount of money to be deposited in the employee's account. Eve can intercept the ciphertext sent to the bank at the end of the month, replace the block with the information about her payment with a copy of the block with the information about the payment of a full-time colleague. Each month Eve can receive more money than she deserves.

Electronic Codebook (ECB)

Error Propagation

A single bit error in transmission can create errors in several in the corresponding block. However, the error <u>does not</u> 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 ← E_K (P_i)
    }
    return Ciphertext blocks
}
```

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
- uses: bulk data encryption, authentication

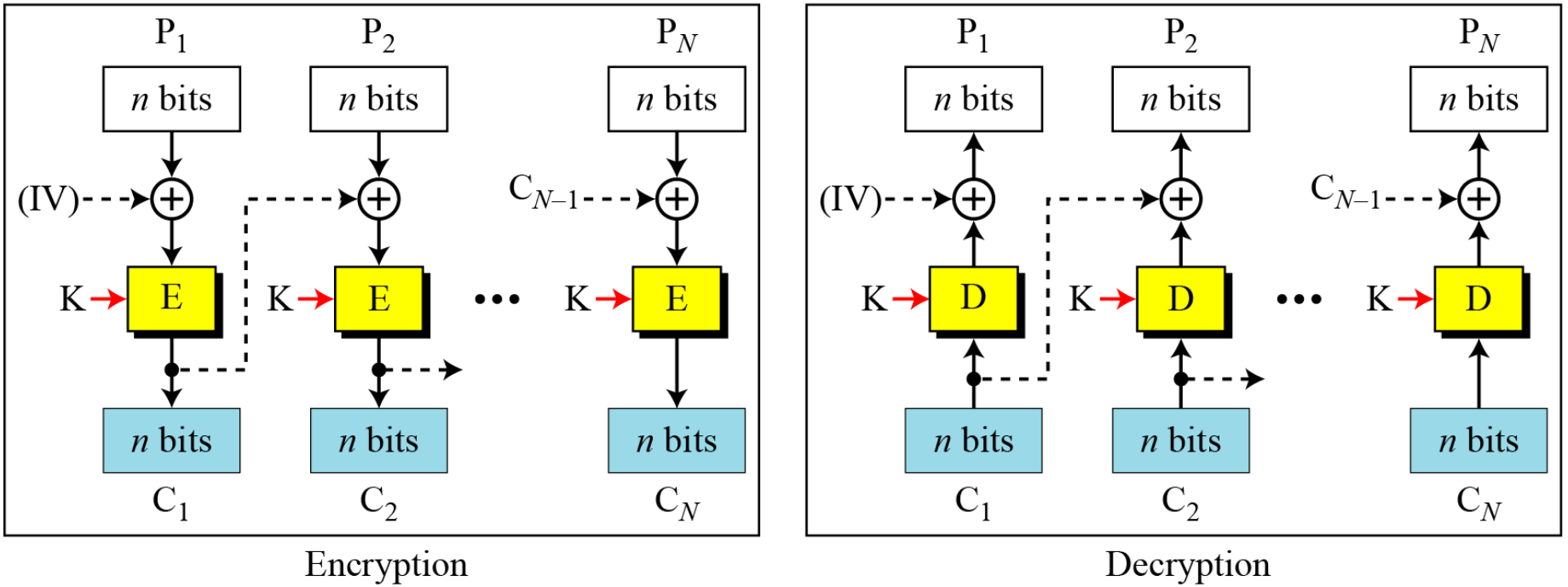
8.1.2 Cipher Block Chaining (CBC) Mode

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

Figure 8.3 *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)

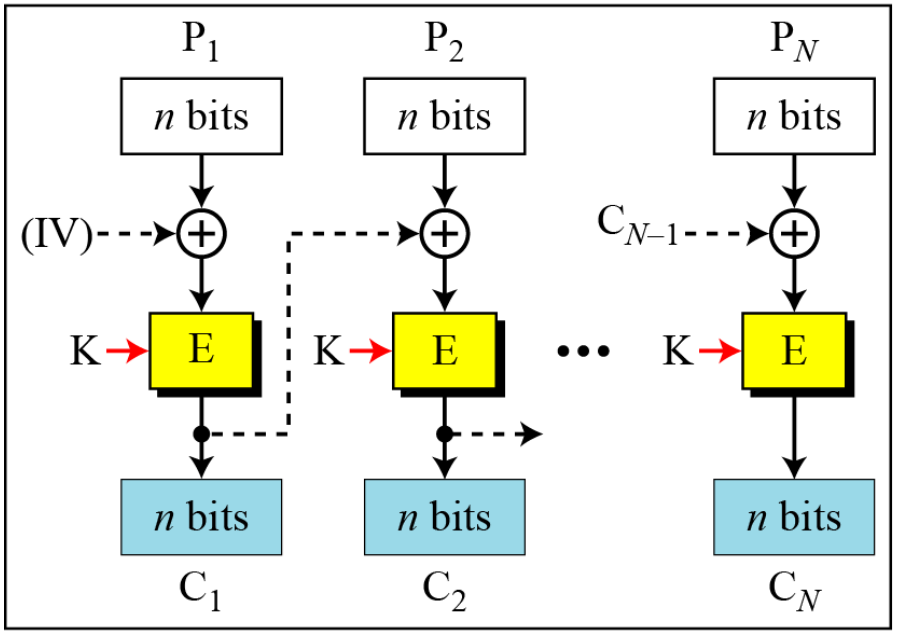


8.1.2 Continued

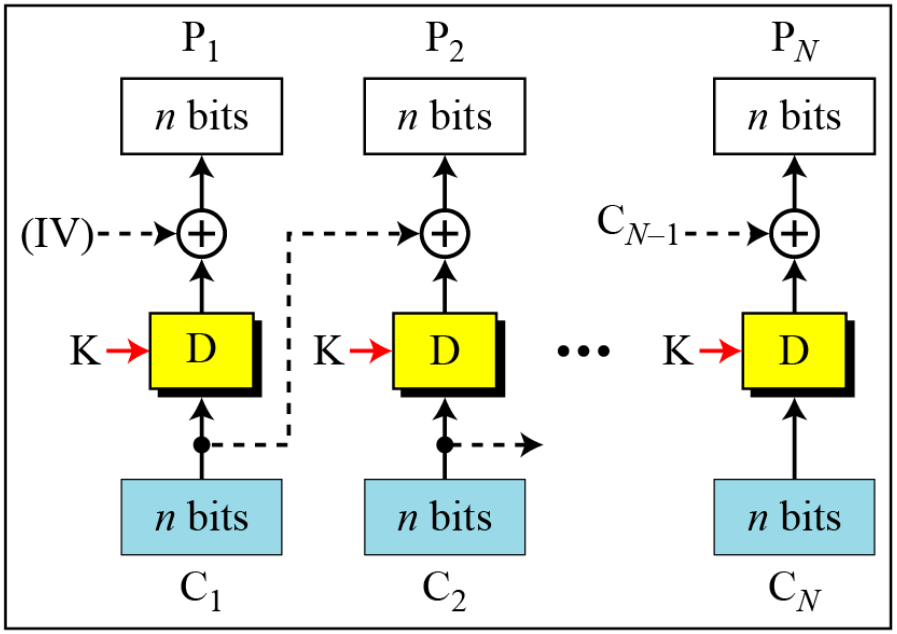
Figure 8.3 *Cipher block chaining (CBC) mode*

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

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



Encryption



Decryption

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

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

Message Padding

- at end of message must handle a possible last short block
  - which is not as large as block size of cipher
  - pad either with known non-data value (e.g. 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

Cipher Feedback (CFB) Mode

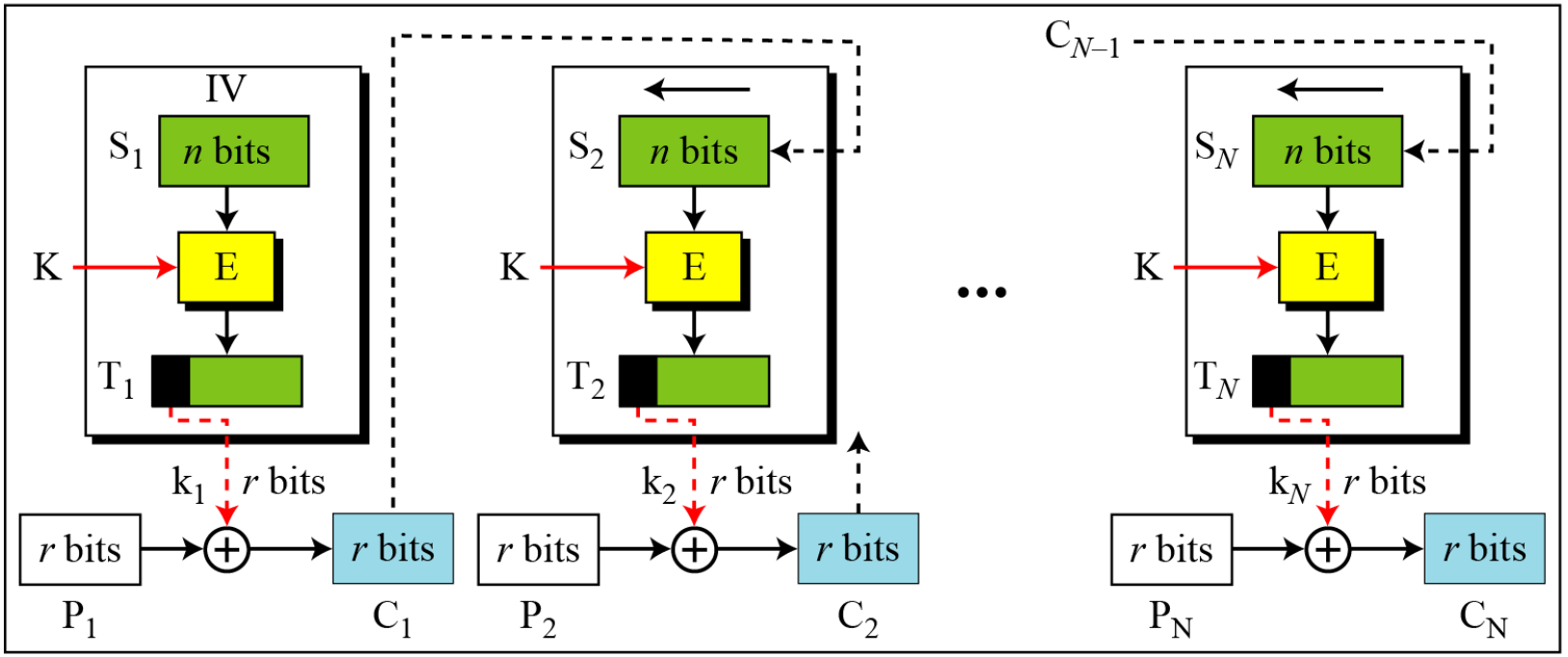
In some situations, we need to use DES or AES as secure ciphers, but the plaintext or ciphertext block sizes are to be smaller.

Figure 8.4 *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



Continued

Note

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

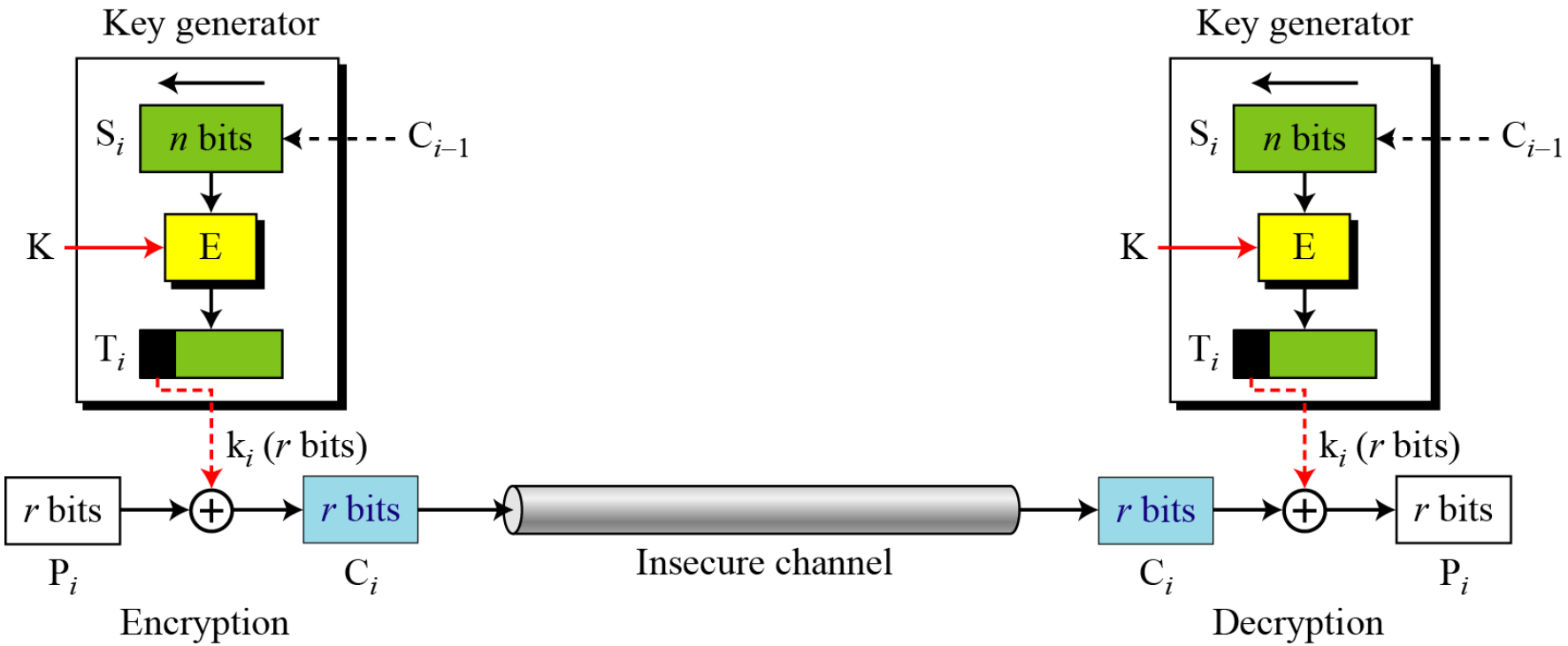
The relation between plaintext and ciphertext blocks is shown below:

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

Continued

CFB as a Stream Cipher

Figure 8.5 *Cipher feedback (CFB) mode as a stream cipher*



Cipher FeedBack (CFB)

- message is treated as a stream of bits
- added to the output of the block cipher
- result is 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)
- uses: stream data encryption, authentication

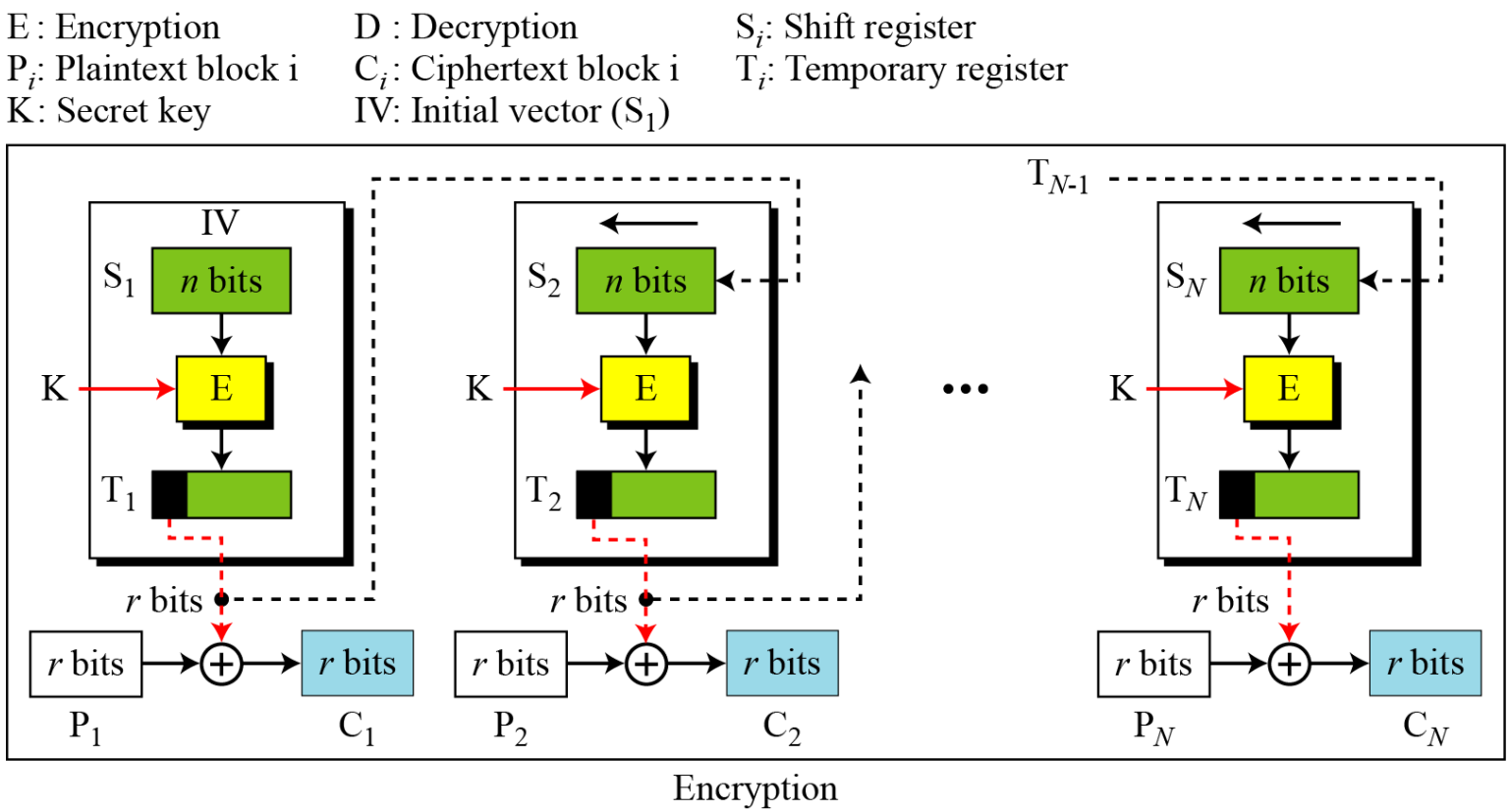
Advantages and Limitations of CFB

- 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) Mode

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

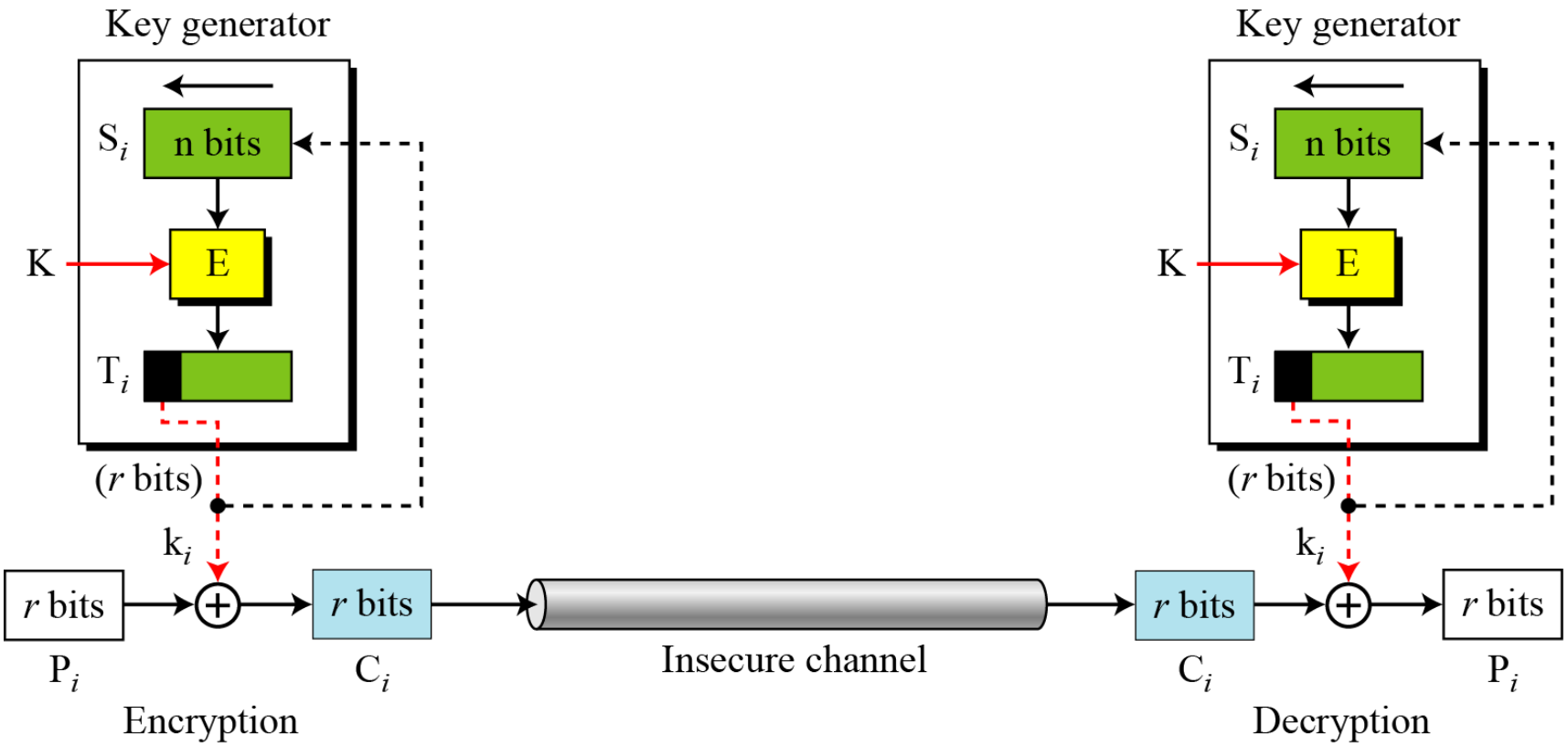
Figure 8.6 *Encryption in output feedback (OFB) mode*



Continued

OFB as a Stream Cipher

Figure 8.7 *Output feedback (OFB) mode as a stream cipher*



Output FeedBack (OFB)

- message is treated as a stream of bits
- output of cipher is added to message
- output is then feed back (hence name)
- feedback is independent of message
- can be computed in advance

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

$$O_i = DES_{K1}(O_{i-1})$$

$$O_{-1} = IV$$

- uses: stream encryption on noisy channels

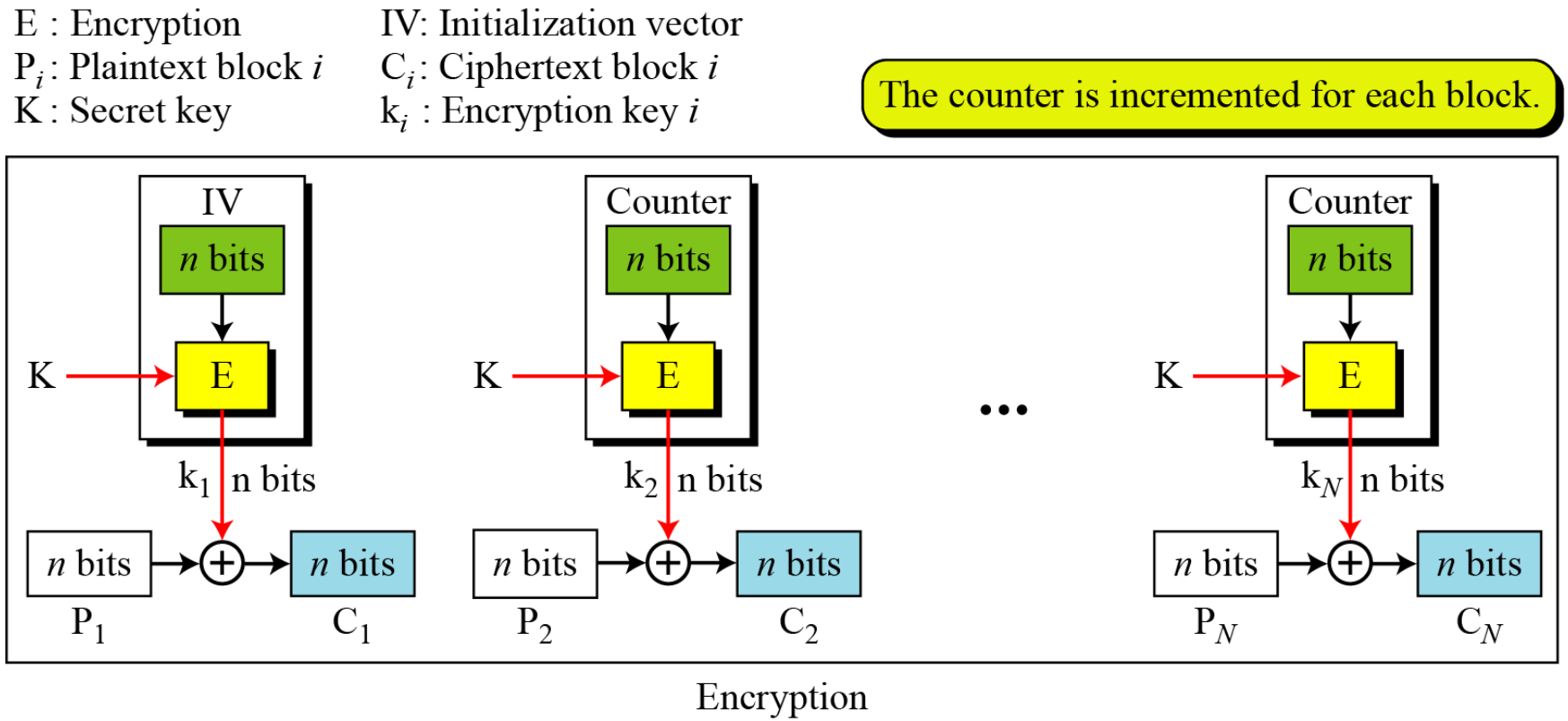
Advantages and Limitations of OFB

- 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 CFB-64 or CFB-128) should ever be used

Counter (CTR) Mode

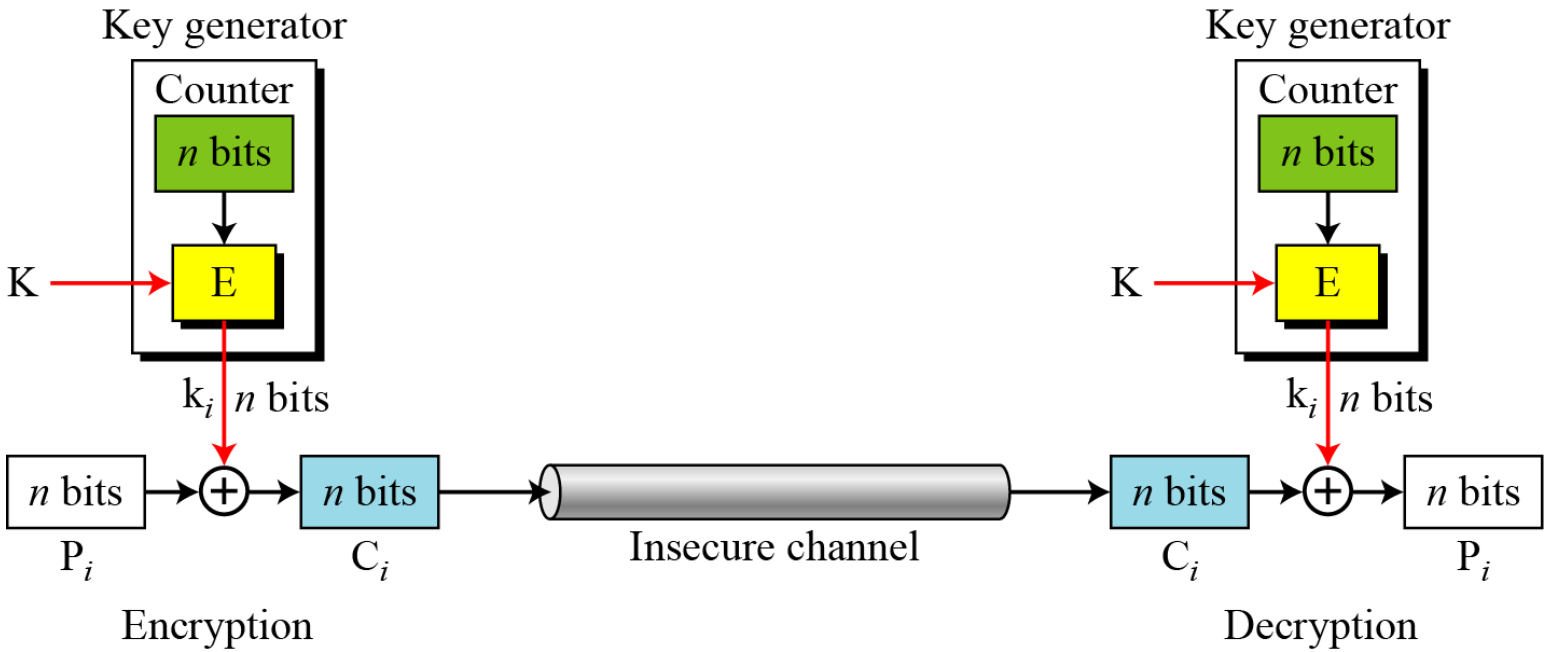
In the counter (CTR) mode, there is no feedback. The pseudorandomness in the key stream is achieved using a counter.

Figure 8.8 *Encryption in counter (CTR) mode*



Continued

Figure 8.9 *Counter (CTR) mode as a stream cipher*



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)

$$O_i = DES_{K1}(i)$$

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

- uses: high-speed network encryptions

Advantages and Limitations of CTR

- efficiency
  - can do parallel encryptions in hardware or software
  - can preprocess in advance of need
  - good for burst 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)

Comparison of Different Modes

Table 8.1 *Summary of operation modes*

Operation Mode	*Description*	*Type of Result*	*Data Unit Size*
ECB	Each *n*-bit block is encrypted independently with the same cipher key.	Block cipher	*n*
CBC	Same as ECB, but each block is first exclusive-ored with the previous ciphertext.	Block cipher	*n*
CFB	Each n-bit block is exclusive-ored with an *r*-bit key, which is part of previous cipher text	Stream cipher	$r \leq n$
OFB	Same as CFB, but the shift register is updated by the previous *r*-bit key.	Stream cipher	$r \leq n$
CTR	Same as OFB, but a counter is used instead of a shift register.	Stream cipher	*n*

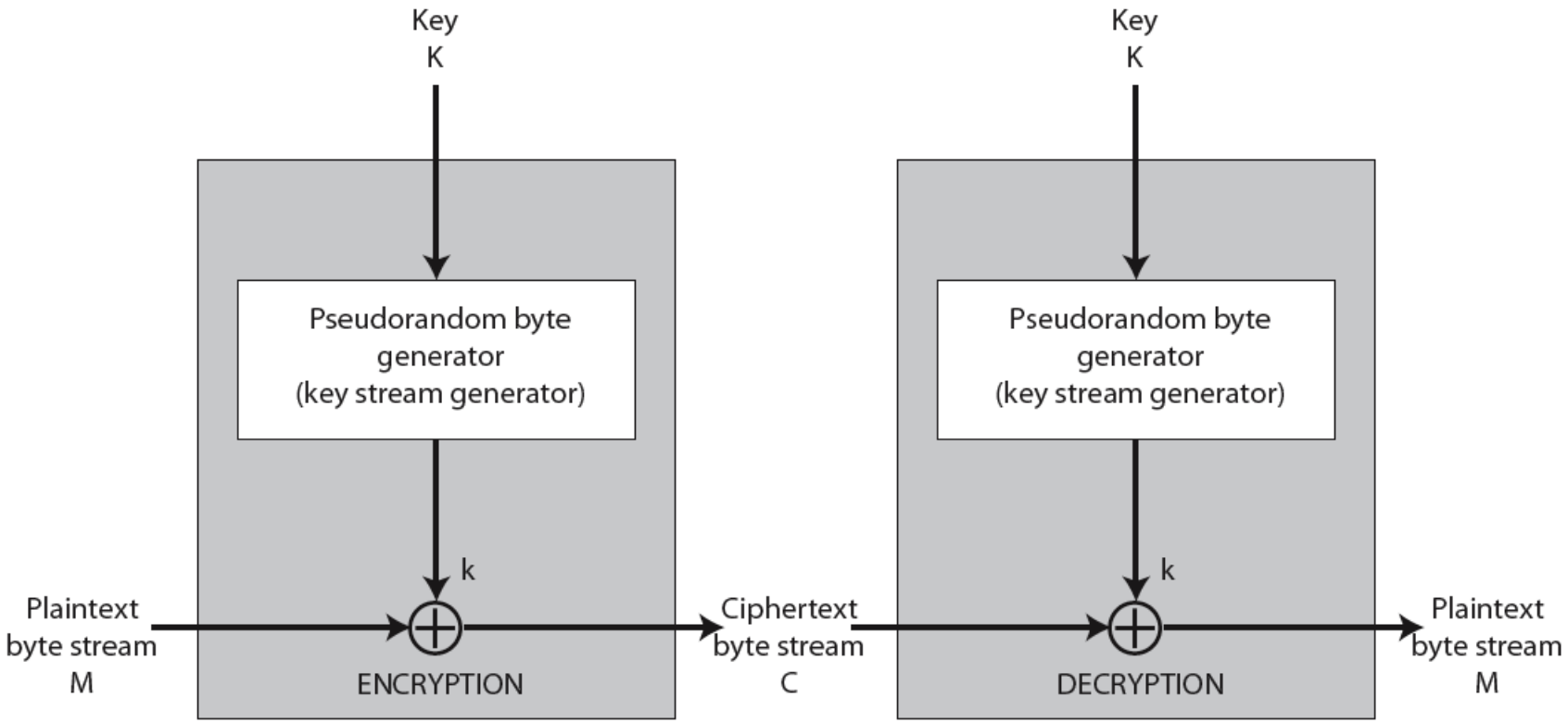
Selection of Modes

- Choice of encryption mode affects
  - Encryption/decryption speed
  - Security against active adversaries (bit flips)
  - Security against passive adversaries (ECB)
  - Error propagation

Stream Ciphers

- process message bit by bit (as a stream)
- have a pseudo random **keystream**
- combined (XOR) with plaintext bit by bit
- randomness of **stream key** completely destroys statistically properties in message
  - $C_i = M_i$ XOR $StreamKey_i$
- but must never reuse stream key
  - otherwise can recover messages (cf book cipher)

Stream Cipher Structure



Stream Cipher Properties

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

RC4

- a proprietary cipher owned by RSA
- another Ron Rivest design, simple but effective
- variable key size, byte-oriented stream cipher
- widely used (web SSL/TLS, wireless WEP)
- key forms random permutation of all 8-bit values
- uses that permutation to scramble input info processed a byte at a time

RC4

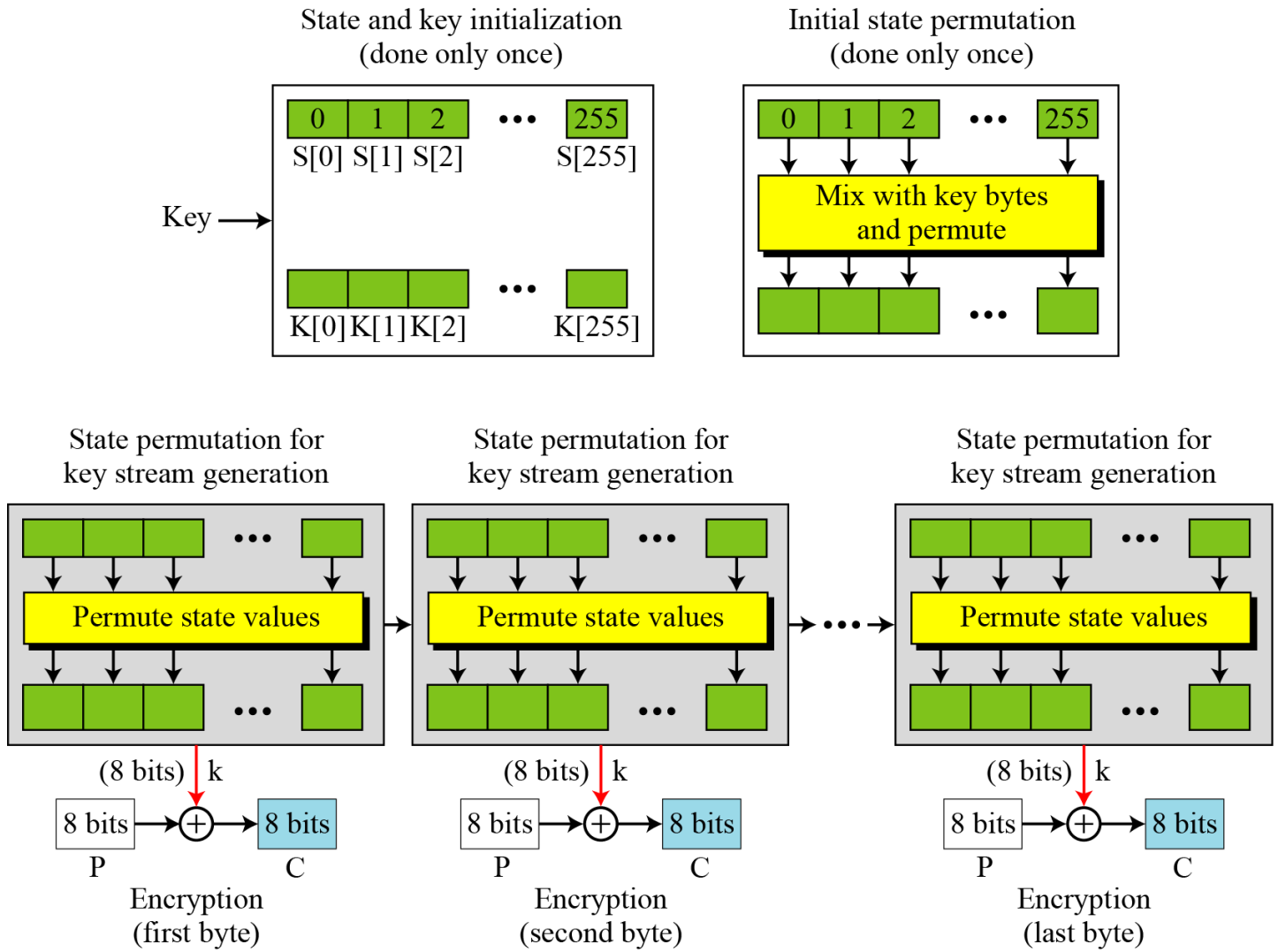
RC4 is a byte-oriented stream cipher in which a byte (8 bits) of a plaintext is exclusive-ored with a byte of key to produce a byte of a ciphertext.

State

RC4 is based on the concept of a state.

S[0] S[1] S[2] ··· S[255]

Figure 8.10 *The idea of RC4 stream cipher*



Continued

Initialization

Initialization is done in two steps:

```
for (i = 0 to 255)
{
    S[i]  ← i
    K[i]  ← Key [i mod KeyLength]
}
```

```
j ← 0
for (i = 0 to 255)
{
    j ← (j + S[i] + K[i]) mod 256
    swap (S[i] , S[j])
}
```

Key Stream Generation

The keys in the key stream are generated, one by one.

```
i  ← (i + 1) mod 256
j  ← (j + S[i]) mod 256
swap (S [i] , S[j])
k  ← S [(S[i] + S[j]) mod 256]
```

Continued

Algorithm

Algorithm 8.6 *Encryption algorithm for RC4*

```
RC4_Encryption (K)
{
    // Creation of initial state and key bytes
    for (i = 0 to 255)
    {
        S[i] ← i
        K[i] ← Key [i mod KeyLength]
    }
    // Permuting state bytes based on values of key bytes
    j ← 0
    for (i = 0 to 255)
    {
        j ← (j + S[i] + K[i]) mod 256
        swap (S[i] , S[j])
    }
```

Continued

Algorithm Continued

```
// Continuously permuting state bytes, generating keys, and encrypting
i ← 0
j ← 0
while (more byte to encrypt)
{
    i ← (i + 1) mod 256
    j ← (j + S[i]) mod 256
    swap (S [i] , S[j])
    k ← S [(S[i] + S[j]) mod 256]
    // Key is ready, encrypt
    input P
    C ← P ⊕ k
    output C
}
}
```

Example 8.5

To show the randomness of the stream key, we use a secret key with all bytes set to 0. The key stream for 20 values of *k* is (222, 24, 137, 65, 163, 55, 93, 58, 138, 6, 30, 103, 87, 110, 146, 109, 199, 26, 127, 163).

Example 8.6

Repeat Example 8.5, but let the secret key be five bytes of (15, 202, 33, 6, 8). The key stream is (248, 184, 102, 54, 212, 237, 186, 133, 51, 238, 108, 106, 103, 214, 39, 242, 30, 34, 144, 49). Again the randomness in the key stream is obvious.

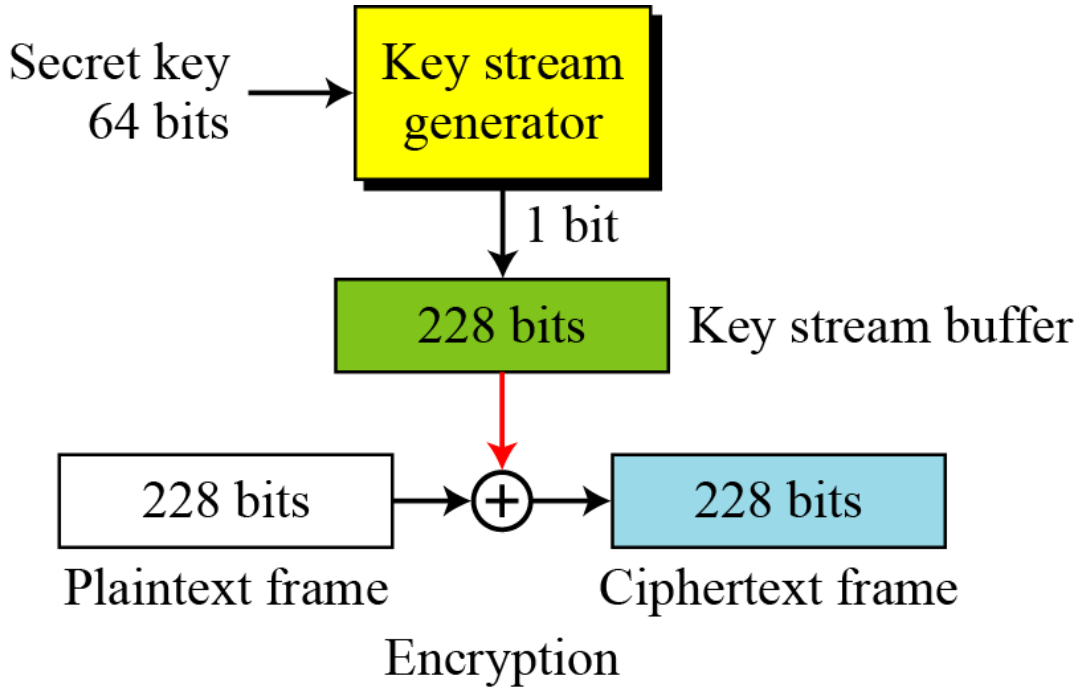
RC4 Security

- claimed secure against known attacks
  - have some analyses, none practical
- result is very non-linear
- since RC4 is a stream cipher, must **never reuse a key**
- have a concern with WEP, but due to key handling rather than RC4 itself

A5/1

A5/1 (a member of the A5 family of ciphers) is used in the Global System for Mobile Communication (GSM), a network for mobile telephone communication..

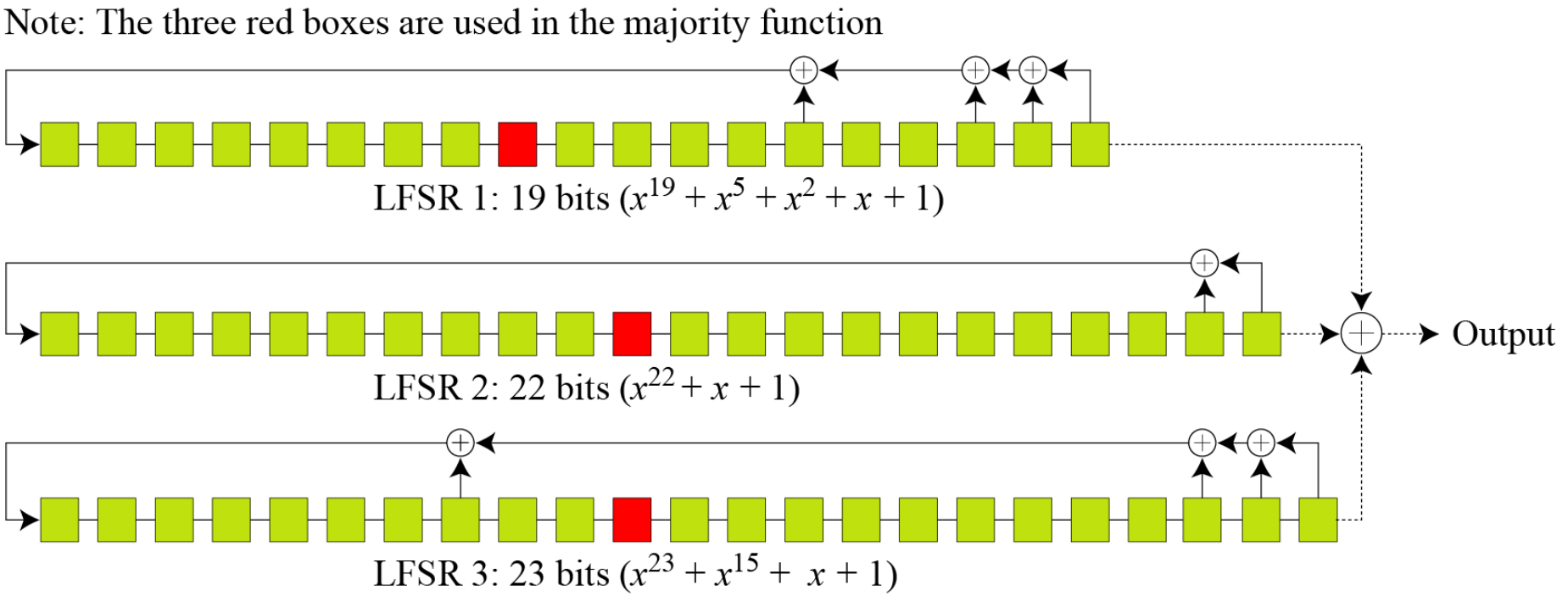
Figure 8.11 *General outline of A5/1*



Key Generator

A5/1 uses three Linear Feedback Shift Registers (LFSRs) with 19, 22, and 23 bits.

Figure 8.12 *Three LFSR's in A5/1*



Continued

Initialization

1. *set all bits in three LFSRs to 0.*
2.

```
for (i = 0 to 63)
   {
      Exclusive-or K[i] with the leftmost bit in all three registers.
      Clock all three LFSRs
   }
```

3.

```
for (i = 0 to 21)
   {
      Exclusive-or FrameNumber [i] with the leftmost bit in all three registers.
      Clock all three LFSRs
   }
```

Continued

4.

```
for (i = 0 to 99)
  {
    Clock the whole generator based on the majority function.
  }
```

Example 8.7

At a point of time the clocking bits are 1, 0, and 1. Which LFSR is clocked (shifted)?

Solution

The result of Majority (1, 0, 1) = 1. LFSR1 and LAFS3 are shifted, but LFSR2 is not.

Encryption/Decryption

The bit streams created from the key generator are buffered to form a 228-bit key that is exclusive-ored with the plaintext frame to create the ciphertext frame. Encryption/decryption is done one frame at a time.

8-3 OTHER ISSUES

Encipherment using symmetric-key block or stream ciphers requires discussion of other issues.

Topics discussed in this section:

8.3.1 Key Management

8.3.2 Key Generation

8.3.1 Key Management

Alice and Bob need to share a secret key between themselves to securely communicate using a symmetric-key cipher. If there are n entities in the community, n(n − 1)/2 keys are needed.

Note

Key management is discussed in Chapter 15.

8.3.2 Key Generation

Different symmetric-key ciphers need keys of different sizes. The selection of the key must be based on a systematic approach to avoid a security leak. The keys need to be chosen randomly. This implies that there is a need for random (or pseudorandom) number generator.

Note

Random number generators are discussed in Appendix K.