More on Symmetric Ciphers

Tsay, Yih-Kuen

Dept. of Information Management

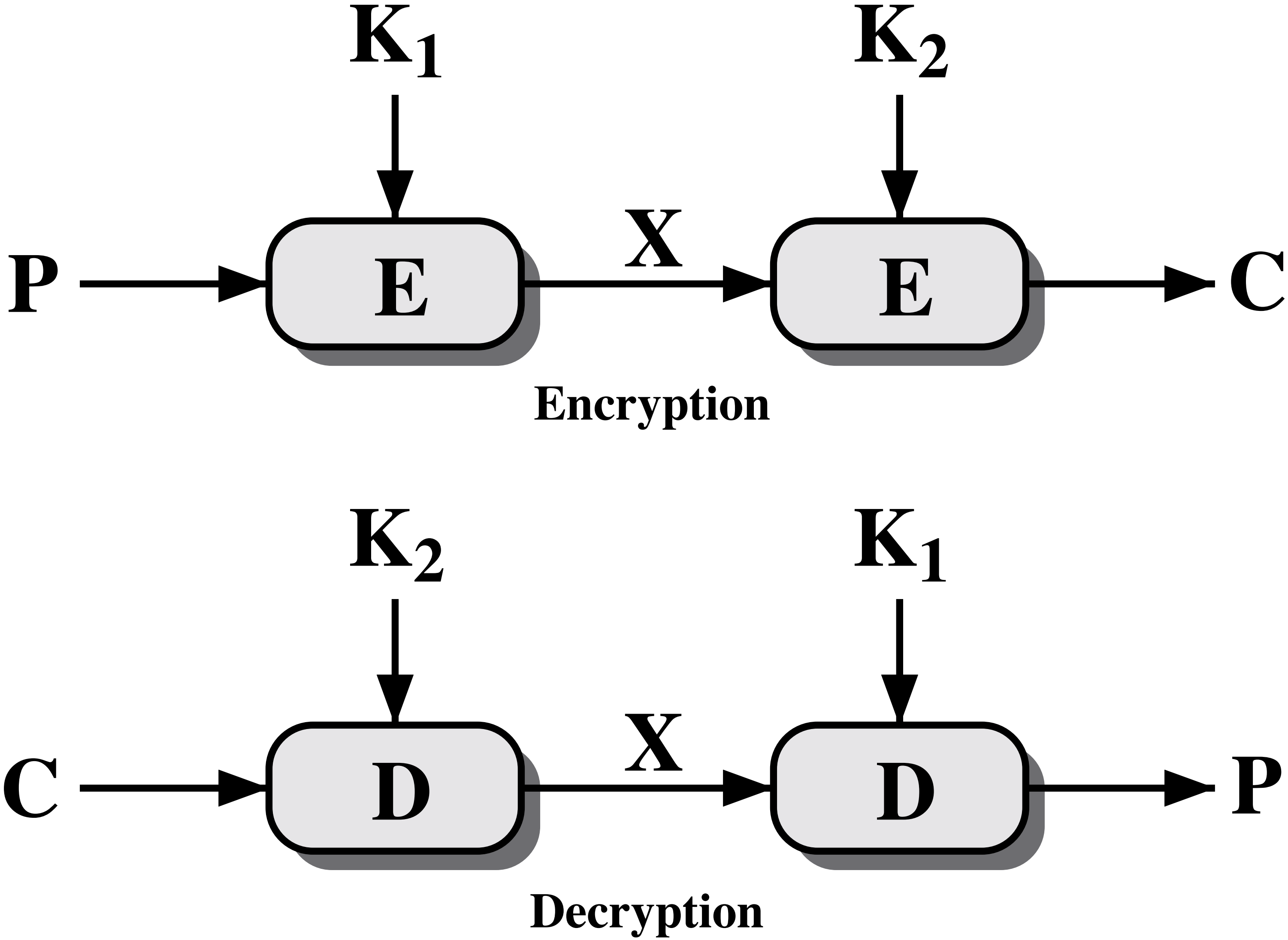
National Taiwan University

Bettering DES

Given the vulnerability of DES to a brute-force attack, there had been (before AES) considerable interest in finding an alternative:

- Completely new algorithms: Blowfish, RC5, ...
- Multiple encryption with DES and multiple keys (to preserve the existing investment in software and equipment):
  - Double DES
  - Triple DES

Multiple Encryption: Double DES



Source: Figure 6.1, Stallings 2006

Reduction to a Single Stage?

- Question: Given any two keys K_1 and K_2, would it be possible to find a key K_3 such that

$$E_{K_2}(E_{K_1}(P)) = E_{K_3}(P)?$$

- If so, then any multiple encryption would be equivalent to some single encryption.
- But, this is unlikely. (Affirmed in 1992.)
  - There are $2^{64}! > 10^{10^{20}}$ distinct permutations of the set of 2^{64} different 64-bit blocks.
  - Each 56-bit DES key defines one such permutation; $2^{56} < 10^{17}$.

Meet-in-the-Middle Attack

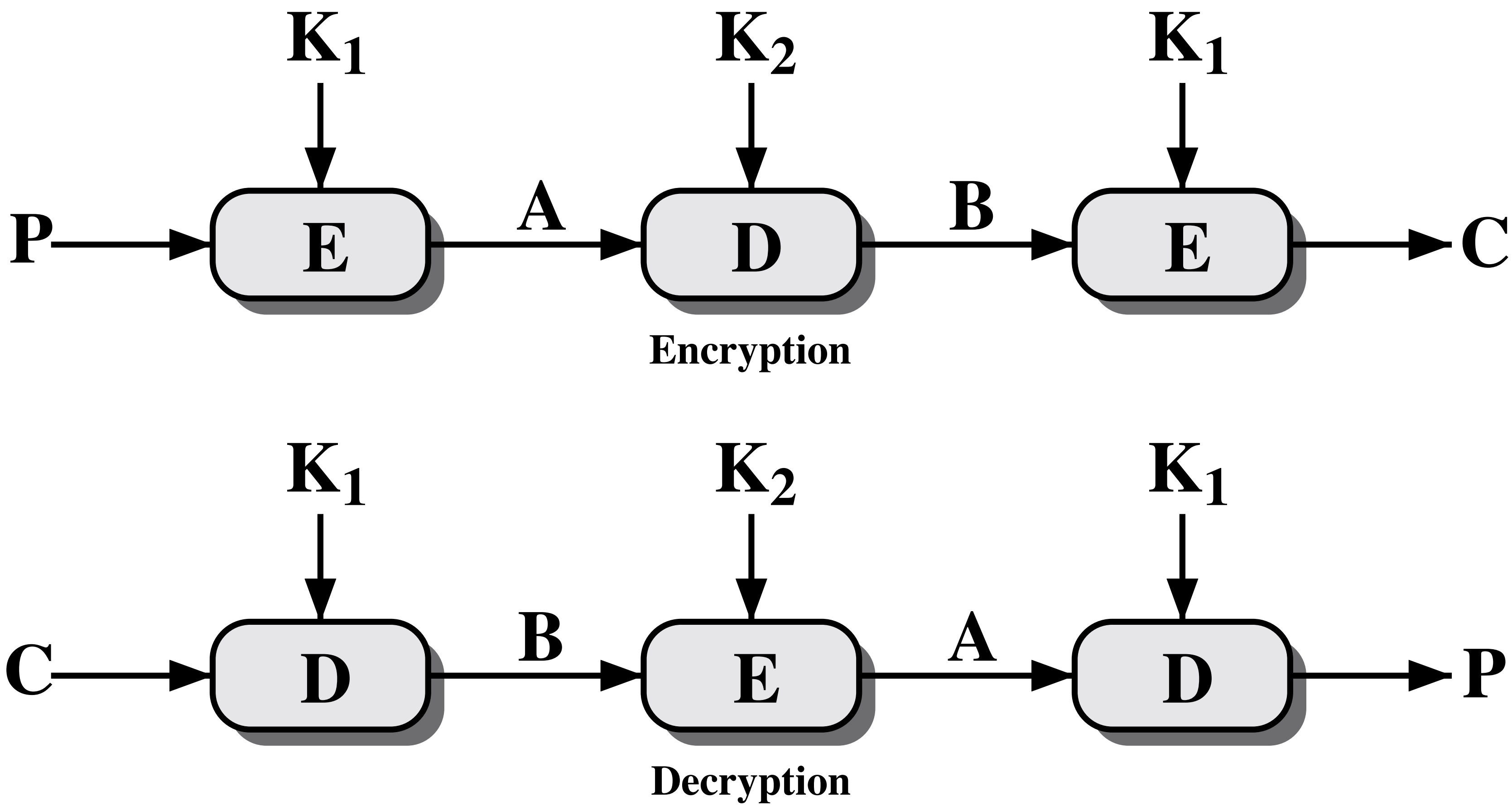
If we have $C = E_{K_2}(E_{K_1}(P))$, then for some X,

$$E_{K_1}(P) = X = D_{K_2}(C)$$

Given a known pair (P, C), the meet-in-the-middle attack proceeds as follows:

1. Encrypt P for all 2^{56} possible values of K_1 and then sort and store the results in a table.
2. Decrypt C using each possible value of K_2 and check the result against the table.
3. If a match occurs, then test the two keys against a new known pair.

Multiple Encryption: Triple DES



Source: Figure 6.1, Stallings 2006

Two-Key Triple DES

- Proposed by Tuchman
- Encryption: $C = E_{K_1}(D_{K_2}(E_{K_1}(P)))$
- Interoperable with DES:

$$E_{K_1}(D_{K_1}(E_{K_1}(P))) = E_{K_1}(P)$$

- Adopted in ANS X9.17, ISO 8732, etc.
- No known practical cryptanalytic attacks

Three-Key Triple DES

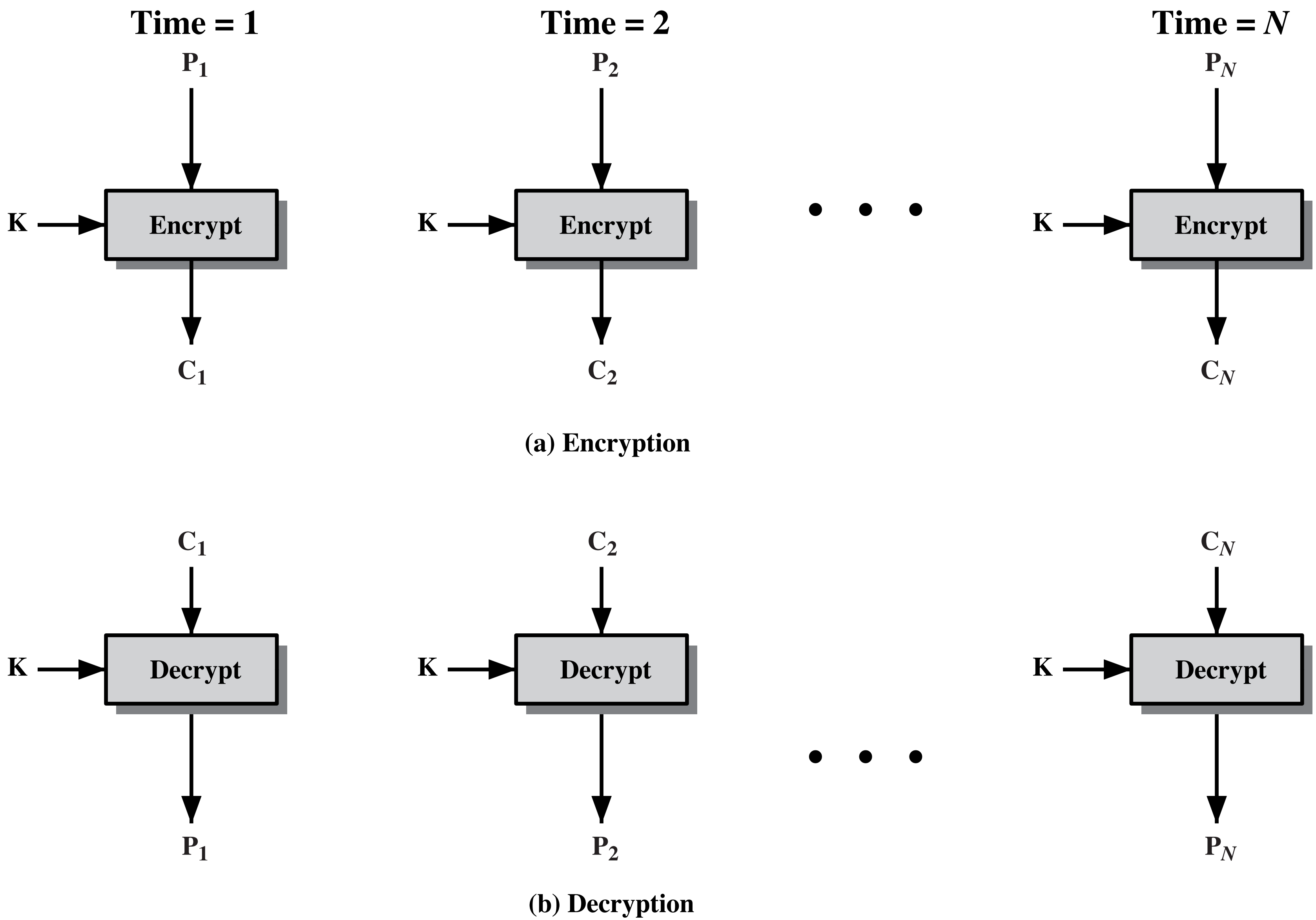
- Many researchers now prefer three-key triple DES
- Encryption: $C = E_{K_3}(D_{K_2}(E_{K_1}(P)))$
- Backward compatible with DES by setting $K_3 = K_2$ or $K_2 = K_1$
- Adopted in PGP, S/MIME, etc.

Modes of Operation

Mode	Description	Typical Application
Electronic Codebook (ECB)	Each block of 64 plaintext bits is encoded independently using the same key.	•Secure transmission of single values (e.g., an encryption key)
Cipher Block Chaining (CBC)	The input to the encryption algorithm is the XOR of the next 64 bits of plaintext and the preceding 64 bits of ciphertext.	•General-purpose block-oriented transmission •Authentication
Cipher Feedback (CFB)	Input is processed *s* bits at a time. Preceding ciphertext is used as input to the encryption algorithm to produce pseudorandom output, which is XORed with plaintext to produce next unit of ciphertext.	•General-purpose stream-oriented transmission •Authentication
Output Feedback (OFB)	Similar to CFB, except that the input to the encryption algorithm is the preceding DES output.	•Stream-oriented transmission over noisy channel (e.g., satellite communication)
Counter (CTR)	Each block of plaintext is XORed with an encrypted counter. The counter is incremented for each subsequent block.	•General-purpose block-oriented transmission •Useful for high-speed requirements

Source: Table 6.1, Stallings 2006

Electronic Codebook (ECB) Mode

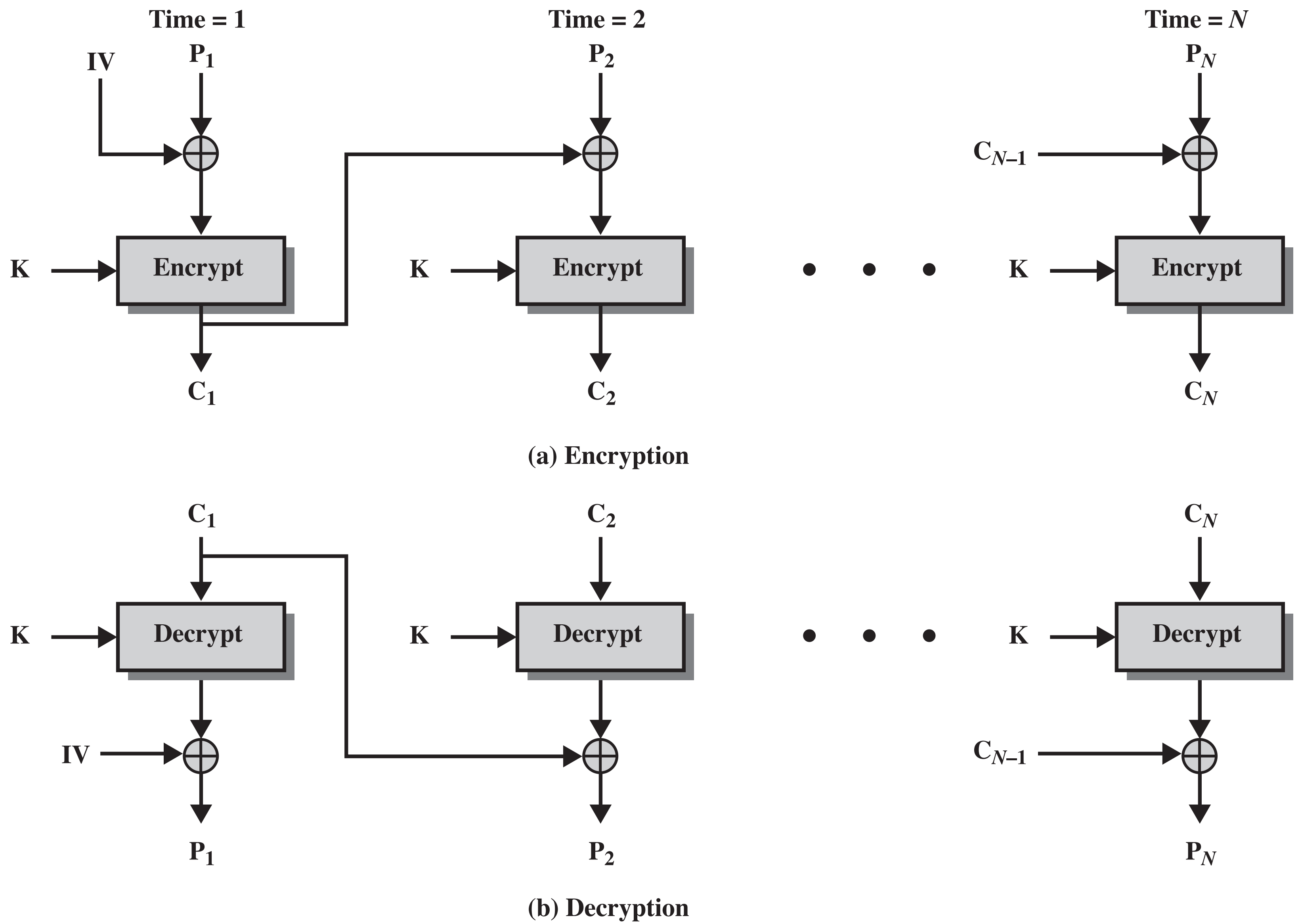


Source: Figure 6.3, Stallings 2006

Characteristics of the ECB Mode

- The same 64-bit block of plaintext produces the same ciphertext
  - May subject the encryption algorithm to known plaintext attacks
  - May be vulnerable to modification attacks (substituting or rearranging blocks)
- Ideal only for a short amount of data such as an encryption key

Cipher Block Chaining (CBC) Mode



Source: Figure 6.4, Stallings 2006

Characteristics of the CBC Mode

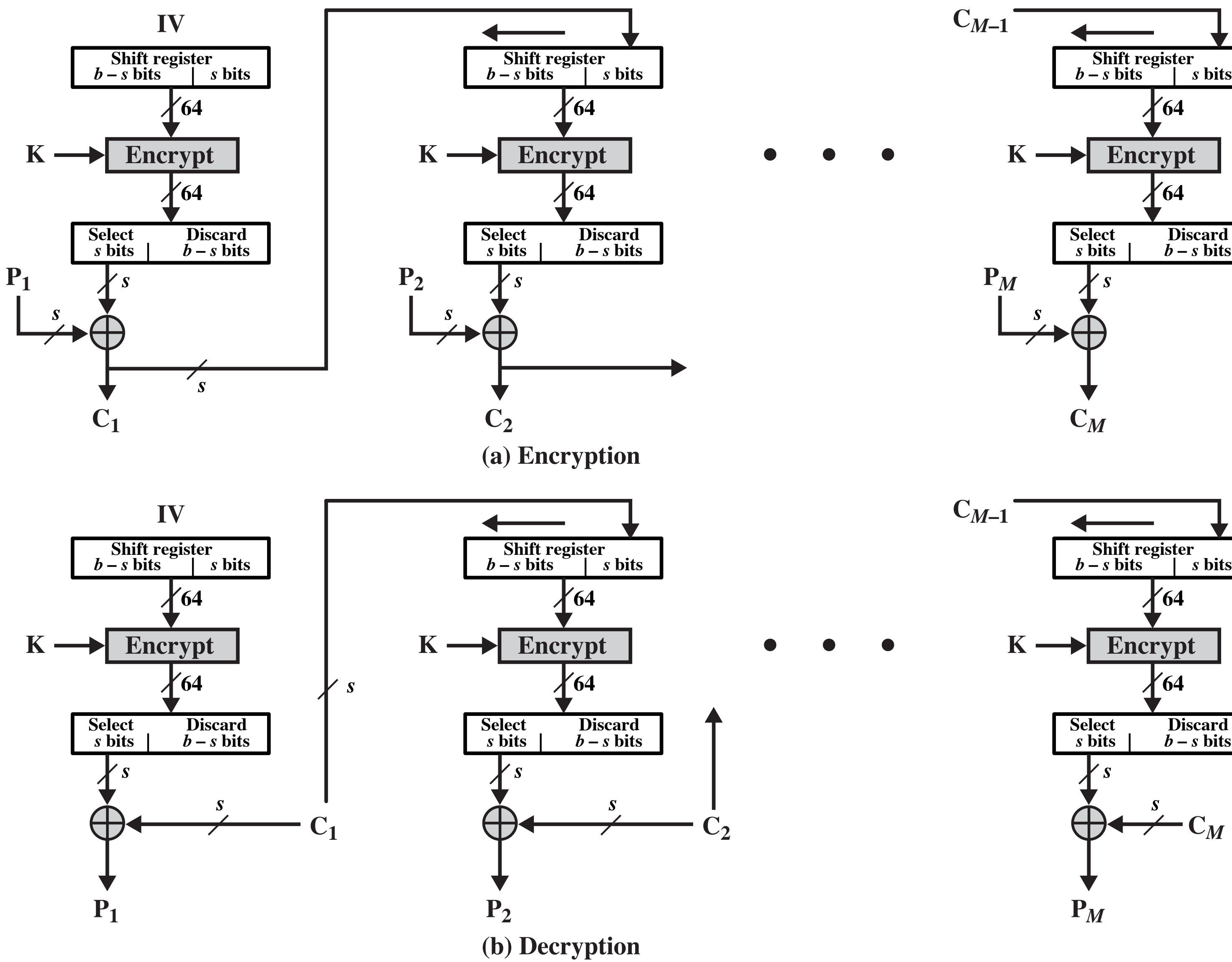
- The Initialization Vector (IV) must be known to both the sender and receiver, and should be protected.
- The opponent may be able to change selected bits of the first block.

$$P_1[i] = IV[i] \oplus D_K(C_1)[i]$$

$$P_1[i]' = IV[i]' \oplus D_K(C_1)[i]$$

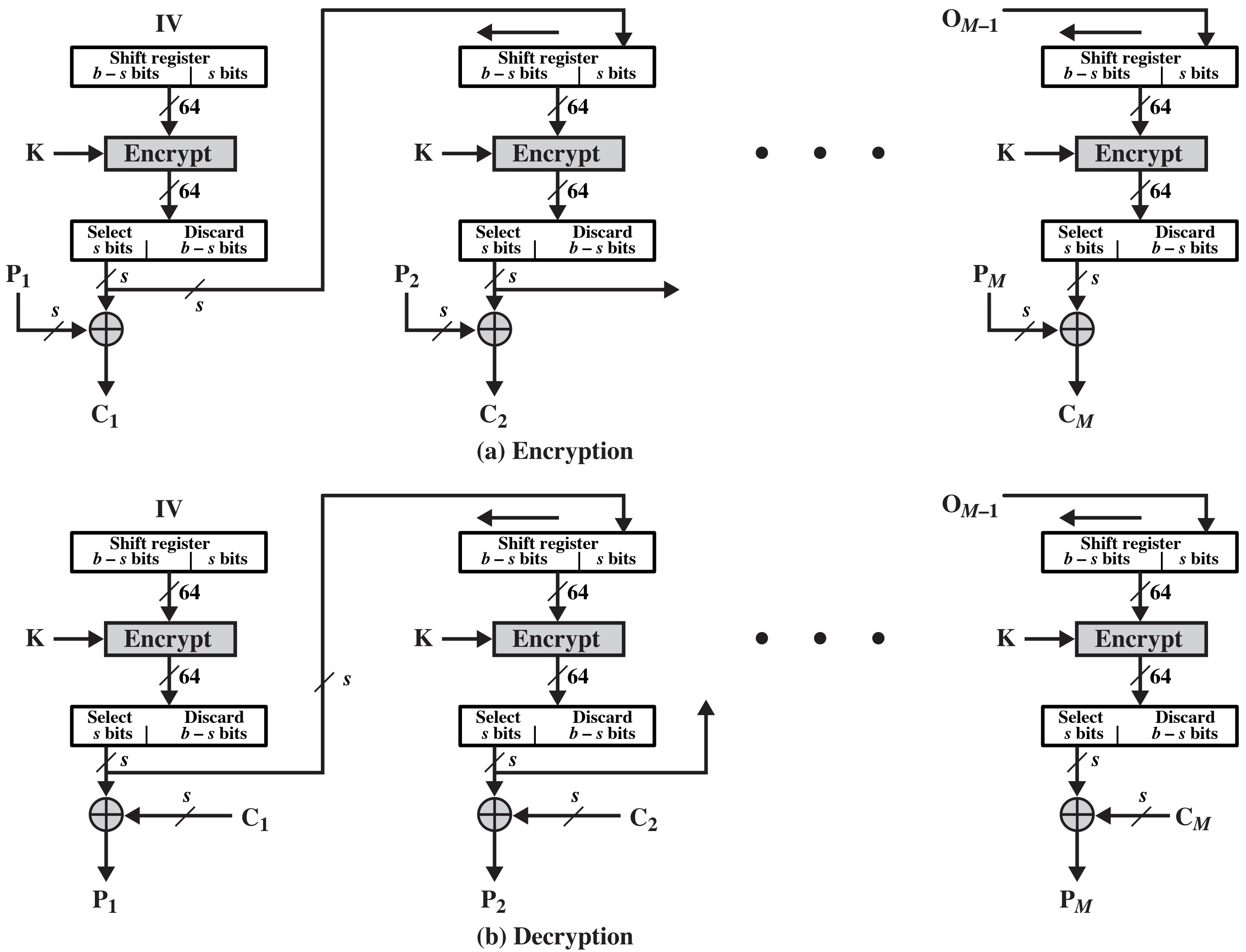
- It can also be used for authentication.

Cipher Feedback (CFB) Mode



Source: Figure 6.5, Stallings 2006

Output Feedback (OFB) Mode

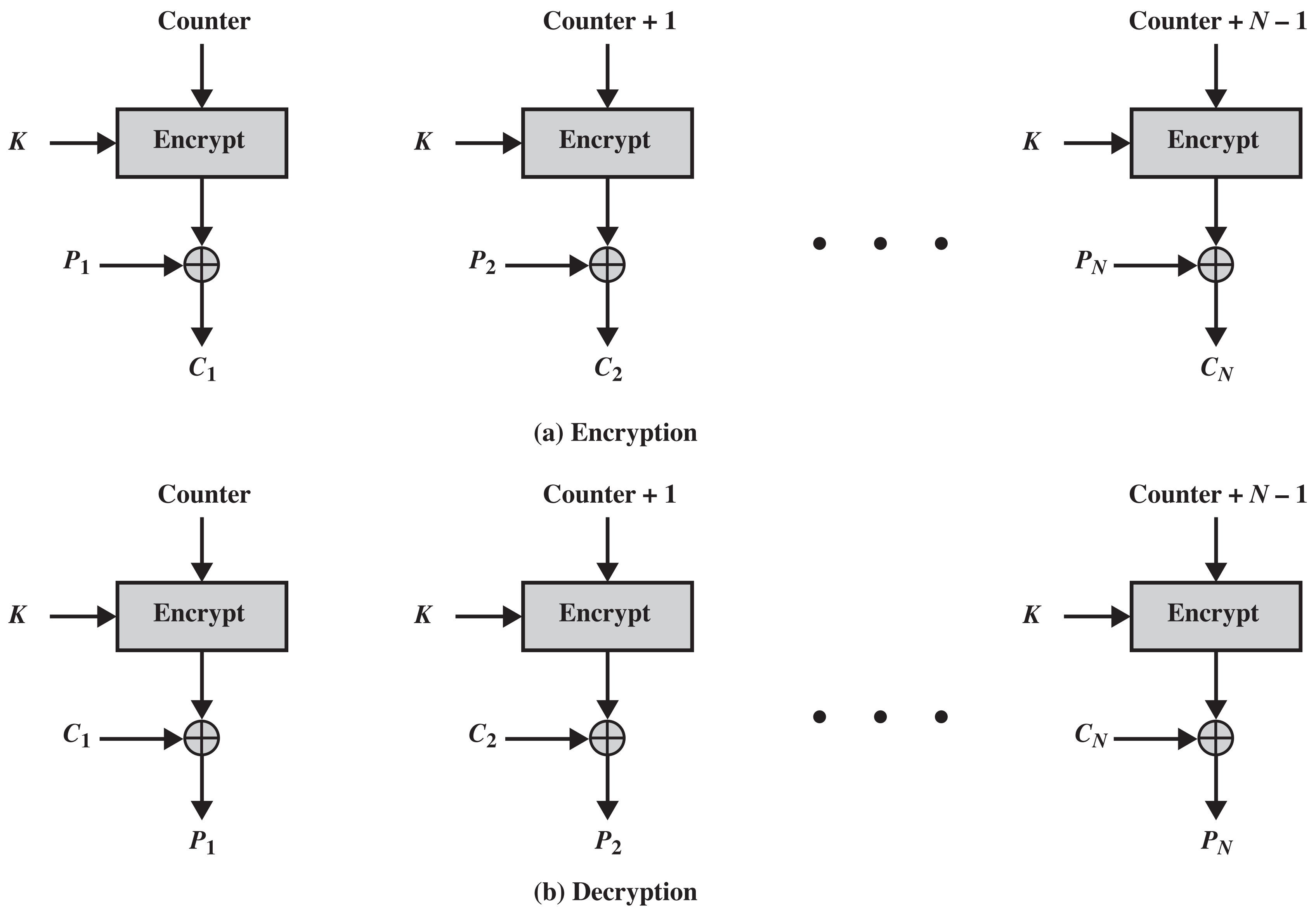


Source: Figure 6.06, Stallings 2006

Characteristics of CFB and OFB

- They both can convert a block cipher into a stream cipher.
- Only the encryption function of a cipher is needed.
- In OFB, bit erros in transmission do not propagate.
- OFB is more vulnerable than CFB to a message stream modification attack.

Counter (CTR) Mode



Source: Figure 6.7, Stallings 2006

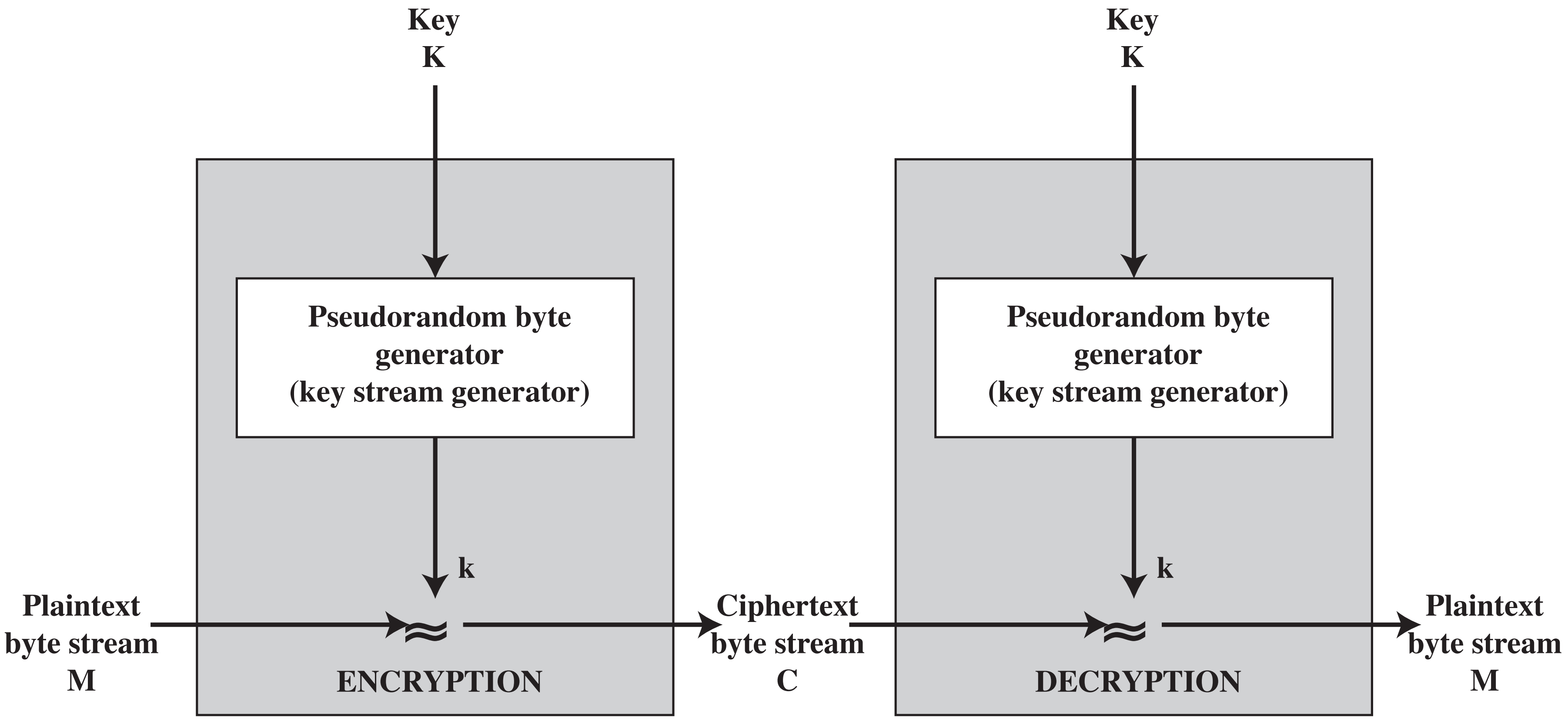
Advantages of the CTR MODE

- Hardware/Software efficiency: parallel processing, pipelining, etc.
- Preprocessing: outputs of the encryption boxes
- Random access
- Provable security: as secure as other modes
- Simplicity: similar to CFB and OFB, only the encryption function is needed

Stream Ciphers

- Encrypt plaintext one byte at a time; other units are possible.
- Typically use a keystream from a pseudorandom byte generator (conditioned on the input key).
- Decryption requires the same pseudorandom sequence.
- Usually are faster and use far less code than block ciphers.
- Design considerations:
  - The encryption sequence should have a large period.
  - The keystream should approximate a truly random stream.
  - The input key needs to be sufficiently long.

Stream Cipher Diagram



Source: Figure 6.8, Stallings 2006

RC4

- Probably the most widely used stream cipher, e.g., in SSL/TLS and in WEP (part of IEEE 802.11)
- Developed in 1987 by Ron Rivest for RSA Security Inc.
- Variable key size with byte-oriented operations
- Based on the use of random permutation
- The period of the cipher likely to be $> 10^{100}$
- Simple and fast
- Proprietary, though its algorithm has been disclosed

Comparisons of Symmetric Ciphers

Cipher	Key Length	Speed (Mbps)
DES	56	9
3DES	168	3
RC2	variable	0.9
RC4	variable	45

Source: Table 6.2, Stallings 2006

Stream Generation in RC4

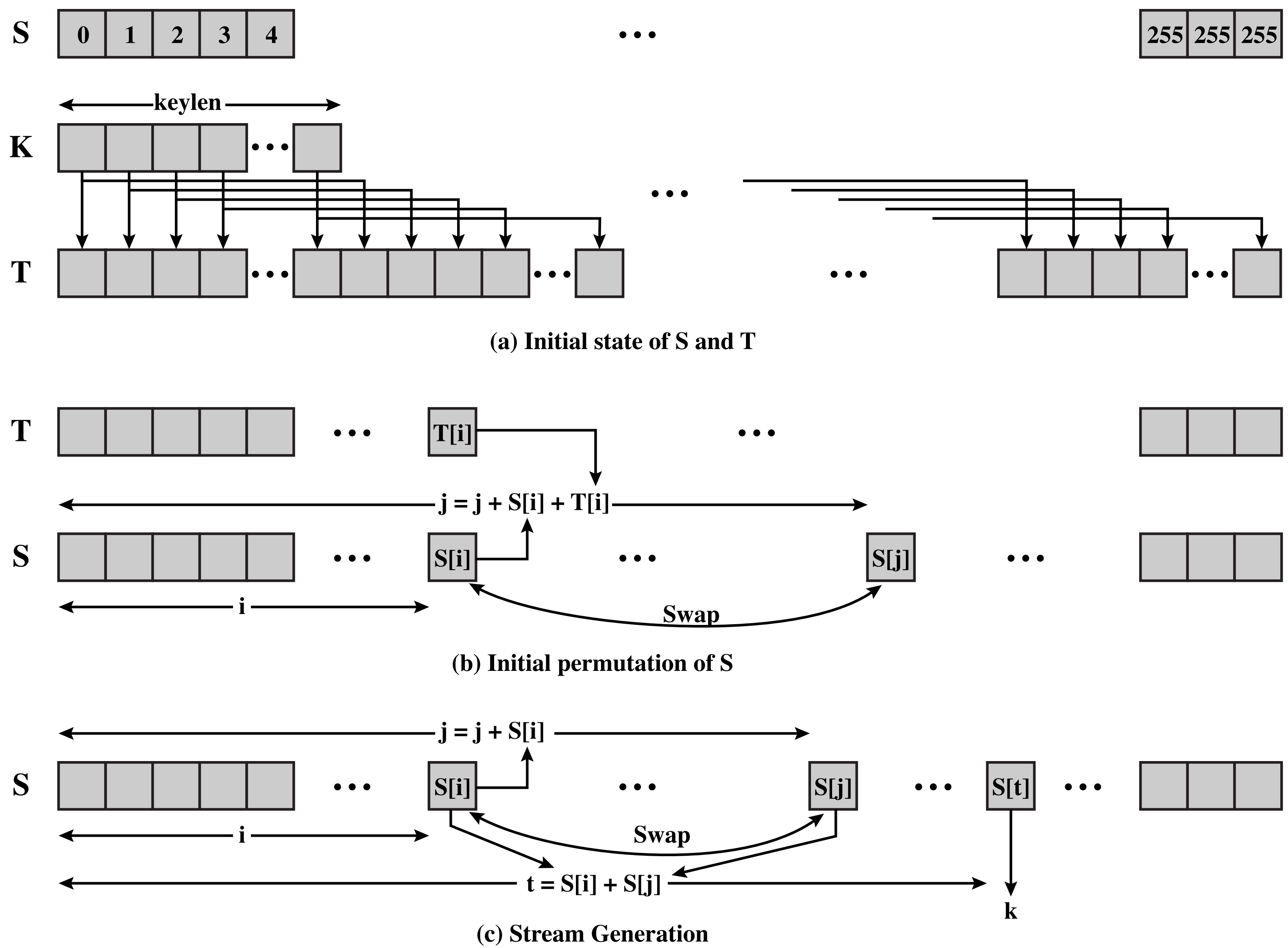
```
i,j = 0;
while (true)
    i = (i + 1) mod 256;
    j = (j + S[i]) mod 256;
    Swap (S[i],S[j]);
    t = (S[i] + S[j]) mod 256;
    k = S[t];
```

Initialization of S in RC4

```
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 in Picture



Source: Figure 6.9, Stallings 2006