CHAPTER 8

Encipherment Using Modern Block Ciphers

(Solution to Odd-Numbered Problems)

Review Questions

1. Modern block ciphers encrypt and decrypt small blocks. DES uses a block size of 8 bytes (characters) and AES uses a block size of 16 bytes (characters). In real life, we need to encrypt or decrypt larger units of data. For example, a one-page document is normally more than 1000 characters. Mode of operations are designed to allow us to repeatedly use a modern block cipher for encryption and decryption.

3. In the electronic codebook (ECB) mode, the plaintext is divided into N blocks. Each block is n bits. The same key is used to encrypt and decrypt each block.

- **Advantages**. This mode has two obvious advantages. First, it is simple. Second, transmission error is not propagated from one block to the other.
- **Disadvantages**. This mode has some security problems. First, patterns at the block level are preserved. Second, block independency creates opportunities for Eve to substitutes some cipher blocks with some cipher blocks of her own.

5. In the cipher feedback (CFB) mode, the size of the plaintext or ciphertext block is r, where $r \leq n$. The idea is to use DES or AES, not for encrypting the plaintext or decrypting the ciphertext, but to encrypt or decrypt the contents of a shift register, S, of size n. Data encryption is done by exclusive-oring an r-bit plaintext block with r bits of the shift register. Data decryption is done by exclusive-oring an r-bit ciphertext block with r bits of the shift register. For each block, the shift register S_i is made by shifting the shift register S_{i-1}(previous shift register) r bits to the left and filling the rightmost r bits with C_{i-1}.

- **Advantages**. One advantage of CFB is that no padding is required because the size of the blocks, r, is normally chosen to fit the data unit to be encrypted. Another advantage is that the system does not have to wait until it has received a large block of data before starting the encryption.
- **Disadvantages.** One disadvantage of CFB is that it is less efficient than CBC or ECB, because it needs to apply the encryption function of underlying block

cipher for each small block of size *r*. Another disadvantage is that Eve can add some ciphertext block to the end of the stream.

7. In the counter (CTR) mode, there is no feedback. The pseudorandomness in the key stream is achieved using a counter. An *n*-bit counter is initialized to a predetermined value (IV) and incremented based on a predefined rule (mod 2^n). To provide a better randomness, the increment value can depend on the block number to be incremented. The plaintext and ciphertext block have the same block size as the underlying cipher (e.g., DEA or AES). Plaintext blocks of size *n* are encrypted to create ciphertext blocks of size *n*.

- **Advantages**. One advantage of CTR is that it can be used to create random access file as long as the value of the counter is related to the record number in the file.
- **Disadvantages.** One disadvantage of CTR is that the size of block is the same as the size of the underlying cipher (DES or AES). Encryption or decryption needs to be done *n*-bit at a time; the process needs to wait until *n* bit of data is accumulated.

9.

First Group: ECB and CBC	**Second Group:** CFB, OFB, and CTR

11. RC4 uses a state of bytes for key generation. Each key in the key stream is a permutation of a key state. A5/1 uses the result of three LFSR's to create a key bit. We can say that key generation in RC4 is byte-oriented, but the key generation in A5/1 is bit-oriented. The other main difference is encryption and decryption. RC4 encrypts and decrypts a character at a time; A5/1 encrypts and decrypts a frame at a time.

13. ECB can be used for parallel processing because each block is encrypted or decrypted independently.

Exercises

15. In CFB mode, the key generator for block *i* uses C_{i-1}, the ciphertext created in block $(i - 1)$. According to our definition in Chapter 5, this is a *nonsynchronous stream cipher*. In OFB mode, the key generator for block *i* uses part of the key from the previous block, but it is independent from the plaintext and ciphertext in the previous block. According to our definition in Chapter 5, this is a *synchronous stream cipher.*

17. In ECB, each block is decrypted independently. However, since the decryption is done a block at a time (DES or AES), the corrupted bit 17 in ciphertext block 8 may affect all *n* bits in plaintext block 8.

19. As discussed in the solution to Exercise 16, a corrupted ciphertext block in CFB affects the corresponding plaintext block and the following n/r plaintext blocks. If

we assume n = 64, the following plaintext blocks may be corrupted: block 11 (bits 3 to 6) and block 12 to 19 (all bits).

21. In OFB mode, the feedback is only in the key-generation system. If ciphertext block 11 is corrupted, only plaintext block 11 may be corrupted.

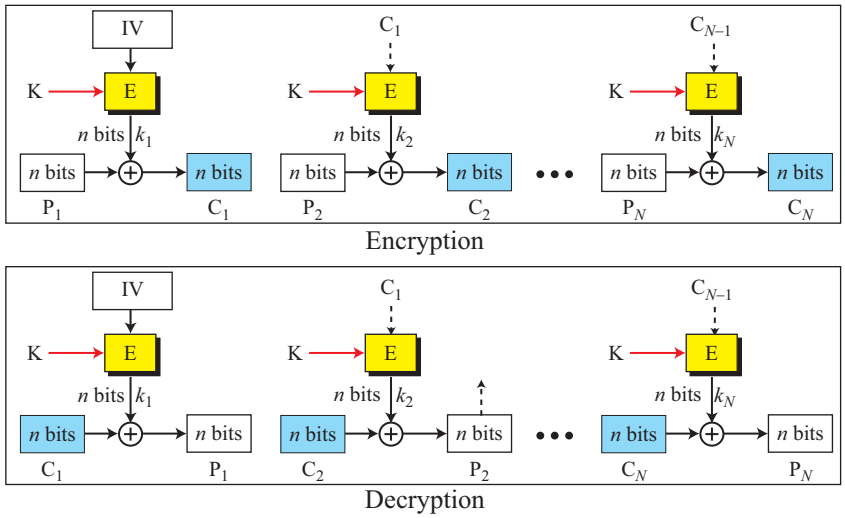
23. In OFB mode, it is obvious that if k_i used at the Bob's site is the same as k_i used at the Alice's site, then $(P_i)'$ created by Bob is the same as P_i sent by Alice (assuming no corruption in transmission):

$$(P_i)' = (C_i)' \otimes k_i = P_i \otimes k_i \otimes k_i = P_i \otimes 0 = P_i$$

The k_i used by Bob is definitely the same as k_i used by Alice if both Alice and Bob use the same IV's.

25. Figure S8.25 shows both the encryption and decryption. It is possible to cancel the underlying encryption for the first block and exclusive-or the IV directly with P_1.

Figure S8.25 *Solution to Exercise 25*



27. The following shows the process:

$$X = D_K(C_{N-1}) \rightarrow P_N = \textbf{head}_m(X)$$
$$Y = C_N \mid \textbf{tail}_{n-m}(X) \rightarrow P_{N-1} = D_K(Y)$$

29. The following shows the process:

$$U = D_K(C_{N-1}) \qquad X = U \oplus [C_N | \textbf{Pad}_{n-m}(0)] \qquad \rightarrow \quad P_N = \textbf{head}_m(X)$$

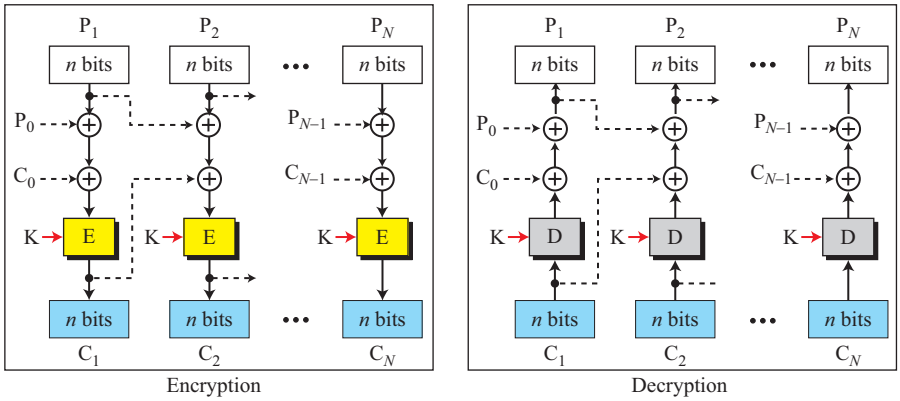
$$Y = DK\,[C_N \mid \textbf{tail}_{n-m}(X)] \qquad \rightarrow \quad P_{N-1} = C_{N-2} \oplus Y$$

31. CFB, OFB, and CRT are stream ciphers, in which the size of the block is usually fixed (one character). There is no need for padding. This means there is no need for cipher stealing technique.

33. In this case, the error propagation is the same as the case where the CTS technique is not used except for the last two blocks. If C_{N-1} is corrupted, it corrupts both P_{N-1} and P_N. If C_N is corrupted, it corrupts only P_{N-1}.

35. Figure S8.35 shows the PCBC mode. In PCBC an error in a ciphertext block corrupts all plaintext block that follows. This mode was used in Kerberos (See Chapter 15) to create both encryption and integrity of blocks. If a block was corrupted, all previous blocks were discarded.

Figure S8.35 *Solution to Exercise 35*



37. We have written a program based on Algorithm 8.6 of the textbook. The result is

41 63 2 212 127 55 201 182 51 242 175 82 133 254 180 107 230 32 241 57

39. This problem can be solved using the classical third birthday problem that we review in Appendix E. We have a sample set of *k* values, in which each sample can take one of the N values. What is the minimum size of *k* such that it is probable (with probability $P \geq 1/2$) such that at least two samples are the same. The answer is $k = 1.18\ N^{1/2}$. In this case $N = 2^{128}$ possible keys, which means $k = 1.18 \times 2^{64}$.

41.

a. Majority (1, 0, 0) = 0. Therefore, $LFSR_2$ and $LFSR_3$ whose clocking bits matches with the value of the Majority function are clocked.

b. Majority (0, 1, 1) = 1. Therefore, $LFSR_2$ and $LFSR_3$ whose clocking bits matches with the value of the Majority function are clocked.

c. Majority (0, 0, 0) = 0. Therefore all three LFSR's are clocked.

d. Majority (1, 1, 1) = 1. Therefore all three LFSR's are clocked.

43.

```
ECB_Decryption (K, C[1 … N])
{
        for (i = 1 to N)
        {
                P[i] ← D_K (C[i])
        }
        return (P[1 … N])
}
```

45. We use a variable Pre_C to hold the previous cipher block. In this way, we do not have to keep an array of ciphertext blocks.

```
CFB_Decryption (IV, K, r)
{
        i ← 1
        while (more blocks to decrypt)
        {
                input (C)
                if (i = 1)
                        S ← IV
                else
                {
                        Temp ← shiftLeft(r , S)
                        S ← concatenate(Temp, Pre_C)
                }
                T ← E_K(S)
                k ← selectLeft(r, T)
                P ← C ⊕ k
                output (P)
                Pre_C ← C
                i ← i + 1
        }
}
```

47. We have written the decryption algorithm assuming all N blocks are ready for decryption to match the encryption algorithm in the text. However, as described in the text, the algorithm can be written if ciphertext blocks are ready only one at a time.

```
CTR_Decryption (IV, K, C[1 … N])
{
        Counter ← IV
        for (i = 1 to N)
        {
                Counter ← (Counter + i  − 1) mod 2^N
                k_i ← E_K (Counter)
                P[i] ← C[i] ⊕ k[i]
        }
        return (P[1 … N])
}
```

49.

```
concatenate (X[1 … n], Y[1 … r])
{
        for (i = 1 to r)
                X[n − r + 1] ← Y[i]
        return X
}
```