Problems

6.1 You want to build a hardware device to do block encryption in the cipher block chaining (CBC) mode using an algorithm stronger than DES. 3DES is a good candidate. Figure 6.10 shows two possibilities, both of which follow from the definition of CBC. Which of the two would you choose:

a. For security?
b. For performance?

Figure 6.10 Use of Triple DES in CBC Mode
6.2 Can you suggest a security improvement to either option in Figure 6.10, using only three DES chips and some number of XOR functions? Assume you are still limited to two keys.

6.3 The Merkle-Hellman attack on 3DES begins by assuming a value of A = 0 (Figure 6.1b). Then, for each of the $2^{30}$ possible values of $K_1$, the plaintext $P$ that produces $A = 0$ is determined. Describe the rest of the algorithm.

6.4 With the ECB mode of DES, if there is an error in a block of the transmitted ciphertext, only the corresponding plaintext block is affected. However, in the CBC mode, this error propagates. For example, an error in the transmitted $C_1$ (Figure 6.4) obviously corrupts $P_1$ and $P_2$.
   a. Are any blocks beyond $P_2$ affected?
   b. Suppose that there is a bit error in the source version of $P_1$. Through how many ciphertext blocks is this error propagated? What is the effect at the receiver?

6.5 If a bit error occurs in the transmission of a ciphertext character in 8-bit CFB mode, how far does the error propagate?

6.6 Fill in the remainder of this table:

<table>
<thead>
<tr>
<th>Mode</th>
<th>Encrypt</th>
<th>Decrypt</th>
</tr>
</thead>
<tbody>
<tr>
<td>ECB</td>
<td>$C_j = E(K, P_j^{\oplus})$ $j = 1,\ldots, N$</td>
<td>$P_j = D(K, C_j)$ $j = 1,\ldots, N$</td>
</tr>
<tr>
<td>CBC</td>
<td>$C_1 = E(K, [P_1^{\oplus} \oplus IV])$</td>
<td>$P_1 = D(K, C_1) \oplus IV$</td>
</tr>
<tr>
<td></td>
<td>$C_j = E(K, [P_j^{\oplus} \oplus C_{j-1}])$ $j = 2,\ldots, N$</td>
<td>$P_j = D(K, C_j) \oplus C_{j-1}$ $j = 2,\ldots, N$</td>
</tr>
<tr>
<td>CFB</td>
<td>OFB</td>
<td>CTR</td>
</tr>
</tbody>
</table>

6.7 CBC-Pad is a block cipher mode of operation used in the RC5 block cipher, but it could be used in any block cipher. CBC-Pad handles plaintext of any length. The ciphertext is longer than the plaintext by at most the size of a single block. Padding is used to assure that the plaintext input is a multiple of the block length. It is assumed that the original plaintext is an integer number of bytes. This plaintext is padded at the end by from 1 to $bb$ bytes, where $bb$ equals the block size in bytes. The pad bytes are all the same and set to a byte that represents the number of bytes of padding. For example, if there are 8 bytes of padding, each byte has the bit pattern 00001000. Why not allow zero bytes of padding? That is, if the original plaintext is an integer multiple of the block size, why not refrain from padding?

6.8 Padding may not always be appropriate. For example, one might wish to store the encrypted data in the same memory buffer that originally contained the plaintext. In that case, the ciphertext must be the same length as the original plaintext. A mode for that purpose is the ciphertext stealing (CTS) mode. Figure 6.11a shows an implementation of this mode.
   a. Explain how it works.
   b. Describe how to decrypt $C_{n-1}$ and $C_n$.

6.9 Figure 6.11b shows an alternative to CTS for producing ciphertext of equal length to the plaintext when the plaintext is not an integer multiple of the block size.
   a. Explain the algorithm.
   b. Explain why CTS is preferable to this approach illustrated in Figure 6.11b.

6.10 What RC4 key value will leave S unchanged during initialization? That is, after the initial permutation of S, the entries of S will be equal to the values from 0 through 255 in ascending order.
6.11 RC4 has a secret internal state which is a permutation of all the possible values of the vector $S$ and the two indices $i$ and $j$.

a. Using a straightforward scheme to store the internal state, how many bits are used?

b. Suppose we think of it from the point of view of how much information is represented by the state. In that case, we need to determine how many states there are, and then take the log to the base 2 to find out how many bits of information this represents. Using this approach, how many bits would be needed to represent the state?

6.12 Alice and Bob agree to communicate privately via email using a scheme based on RC4, but want to avoid using a new secret key for each transmission. Alice and Bob privately agree on a 128-bit key $k$. To encrypt a message $m$, consisting of a string of bits, the following procedure is used:

1. Choose a random 80-bit value $v$
2. Generate the ciphertext $c = RC4(v || k) \oplus m$
3. Send the bit string $(v \parallel c)$

a. Suppose Alice uses this procedure to send a message $m$ to Bob. Describe how Bob can recover the message $m$ from $(v \parallel c)$ using $k$.

b. If an adversary observes several values $(v_1 \parallel c_1), (v_2 \parallel c_2), \ldots$ transmitted between Alice and Bob, how can he/she determine when the same key stream has been used to encrypt two messages?
c. Approximately how many messages can Alice expect to send before the same key stream will be used twice? Use the result from the birthday paradox described in Appendix 11A [Equation (11.7)].

d. What does this imply about the lifetime of the key k (i.e., the number of messages that can be encrypted using k)?

Programming Problems

6.13 Create software that can encrypt and decrypt in Cipher Block Chaining mode using one of the following ciphers: affine modulo 256, Hill modulo 256, S-DES, DES. Test data for S-DES: using a binary initialization vector of 1010 1010, a binary plaintext of 0000 0001 0010 0011 encrypted with a binary key of 01111 11101 should give a binary plaintext of 1111 0100 0000 1011. Decryption should work correspondingly.

6.14 Create software that can encrypt and decrypt in 4-bit Cipher Feedback mode using one of the following ciphers: additive modulo 256, affine modulo 256, S-DES;

   or

8-bit Cipher Feedback mode using one of the following ciphers: $2 \times 2$ Hill modulo 256. Test data for S-DES: using a binary initialization vector of 1010 1011, a binary plaintext of 0001 0010 0011 0100 encrypted with a binary key of 01111 11101 should give a binary plaintext of 1110 1100 1111 1010. Decryption should work correspondingly.

6.15 Create software that can encrypt and decrypt in 4-bit Output Feedback mode using one of the following ciphers: additive modulo 256, affine modulo 256, S-DES;

   or

8-bit Output Feedback mode using one of the following ciphers: $2 \times 2$ Hill modulo 256,

6.16 Create software that can encrypt and decrypt in Counter mode using one of the following ciphers: affine modulo 256, Hill modulo 256, S-DES. Test data for S-DES: using a counter starting at 0000 0000, a binary plaintext of 0000 0001 0000 0010 0000 0100 encrypted with a binary key of 01111 11101 should give a binary plaintext of 0011 1000 0100 1111 0011 0010. Decryption should work correspondingly.

6.17 Implement a differential cryptanalysis attack on 3-round S-DES.