Chapter 6 Block Ciphers

6.1 Block Ciphers

In many classical cryptosystems, changing one letter in the plaintext changes exactly one letter in the ciphertext. In the shift, affine, and substitution ciphers, a given letter in the ciphertext always comes from exactly one letter in the plaintext. This greatly facilitates finding the key using frequency analysis. In the Vigenère system, the use of blocks of letters, corresponding to the length of the key, makes the frequency analysis more difficult, but still possible, since there is no interaction among the various letters in each block. Block ciphers avoid these problems by encrypting blocks of several letters or numbers simultaneously. A change of one character in a plaintext block should change potentially all the characters in the corresponding ciphertext block.

The Playfair cipher in Section 2.6 is a simple example of a block cipher, since it takes two-letter blocks and encrypts them to two-letter blocks. A change of one letter of a plaintext pair always changes at least one letter, and usually both letters, of the ciphertext pair. However, blocks of two letters are too small to be secure, and frequency analysis, for example, is usually successful.

Many of the modern cryptosystems that will be treated later in this book are block ciphers. For example, DES operates on blocks of 64 bits. AES uses blocks of 128 bits. RSA sometimes uses blocks more than 1000 bits long, depending on the modulus used. All of these block lengths are long enough to be secure against attacks such as frequency analysis.

Claude Shannon, in one of the fundamental papers on the theoretical foundations of cryptography [Shannon1], gave two properties that a good cryptosystem should have in order to hinder statistical analysis: **diffusion** and **confusion**.

Diffusion means that if we change a character of the plaintext, then several characters of the ciphertext should change, and, similarly, if we change a character of the ciphertext, then several characters of the plaintext should change. This means that frequency statistics of letters, digrams, etc. in the plaintext are diffused over several characters in the ciphertext, which means that much more ciphertext is needed to do a meaningful statistical attack.

Confusion means that the key does not relate in a simple way to the ciphertext. In particular, each character of the ciphertext should depend on several parts of the key. When a situation like this happens, the cryptanalyst probably needs to solve for the entire key simultaneously, rather than piece by piece.

The Vigenère and substitution ciphers do not have the properties of diffusion and confusion, which is why they are so susceptible to frequency analysis.

The concepts of diffusion and confusion play a role in any well-designed block cipher. Of course, a disadvantage (which is precisely the cryptographic advantage) of diffusion is error propagation: A small error in the ciphertext becomes a major error in the decrypted message, and usually means the decryption is unreadable.

The natural way of using a block cipher is to convert blocks of plaintext to blocks of ciphertext, independently and one at a time. This is called the electronic codebook (ECB) mode. Although it seems like the obvious way to implement a block cipher, we'll see that it is insecure and

that there are much better ways to use a block cipher. For example, it is possible to use feedback from the blocks of ciphertext in the encryption of subsequent blocks of plaintext. This leads to the cipher block chaining (CBC) mode and cipher feedback (CFB) mode of operation. These are discussed in Section 6.3.

For an extensive discussion of block ciphers, see [Schneier].

6.2 Hill Ciphers

This section is not needed for understanding the rest of the chapter. It is included as an example of a block cipher.

In this section, we discuss the Hill cipher, which is a block cipher invented in 1929 by Lester Hill. It seems never to have been used much in practice. Its significance is that it was perhaps the first time that algebraic methods (linear algebra, modular arithmetic) were used in cryptography in an essential way. As we'll see in later chapters, algebraic methods now occupy a central position in the subject.

Choose an integer n, for example $n = 3$. The key is an $n \times n$ matrix M whose entries are integers mod 26. For example, let

$$M = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 11 & 9 & 8 \end{pmatrix}.$$

The message is written as a series of row vectors. For example, if the message is *abc*, we change this to the single row vector $(0, 1, 2)$. To encrypt, multiply the vector by the matrix (traditionally, the matrix appears on the right in the multiplication; multiplying on the left would yield a similar theory) and reduce mod 26:

$$(0,\ 1,\ 2) \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 11 & 9 & 8 \end{pmatrix} \equiv (0,\ 23,\ 22) \pmod{26}.$$

Therefore, the ciphertext is *AXW*. (The fact that the first letter a remained unchanged is a random occurrence; it is not a defect of the method.)

In order to decrypt, we need the determinant of M to satisfy

$$\gcd(\det(M),\ 26) = 1.$$

This means that there is a matrix N with integer entries such that $MN \equiv I \pmod{26}$, where I is the $n \times n$ identity matrix.

In our example, $\det(M) = -3$. The inverse of M is

$$\frac{-1}{3}\begin{pmatrix} -14 & 11 & -3 \\ 34 & -25 & 6 \\ -19 & 13 & -3 \end{pmatrix}.$$

Since 17 is the inverse of -3 mod 26, we replace $-1/3$ by 17 and reduce mod 26 to obtain

$$N = \begin{pmatrix} 22 & 5 & 1 \\ 6 & 17 & 24 \\ 15 & 13 & 1 \end{pmatrix}.$$

The reader can check that $MN \equiv I \pmod{26}$.

For more on finding inverses of matrices mod n, see Section 3.8. See also Example 15 in the Computer Appendices.

The decryption is accomplished by multiplying by N, as follows:

$$(0,\ 23,\ 22)\begin{pmatrix} 22 & 5 & 1 \\ 6 & 17 & 24 \\ 15 & 13 & 1 \end{pmatrix} \equiv (0,\ 1,\ 2) \pmod{26}$$

In the general method with an $n \times n$ matrix, break the plaintext into blocks of n characters and change each block to a vector of n integers between 0 and 25 using $a = 0, b = 1, \ldots, z = 25$. For example, with the matrix M as above, suppose our plaintext is

blockcipher.

This becomes (we add an x to fill the last space)

$$1 \quad 11 \quad 14 \qquad 2 \quad 10 \quad 2 \qquad 8 \quad 15 \quad 7 \qquad 4 \quad 17 \quad 23.$$

Now multiply each vector by M, reduce the answer mod 26, and change back to letters:

$$\begin{aligned}(1, 11, 14)M &= (199, 183, 181) \equiv (17,\ 1,\ 25) \pmod{26} = RBZ\\ (2, 10,\ 2)M &= (64, 72, 82) \equiv (12,\ 20,\ 4) \pmod{26} = MUE,\\ &\text{etc.}\end{aligned}$$

In our case, the ciphertext is

$$RBZMUEPYONOM.$$

It is easy to see that changing one letter of plaintext will usually change n letters of ciphertext. For example, if *block* is changed to *clock*, the first three letters of ciphertext change from RBZ to SDC. This makes frequency counts less effective, though they are not impossible when n is small. The frequencies of two-letter combinations, called **digrams**, and three-letter combinations, **trigrams**, have been computed. Beyond that, the number of combinations becomes too large (though tabulating the results for certain common combinations would not be difficult). Also, the frequencies of combinations are so low that it is hard to get meaningful data without a very large amount of text.

Now that we have the ciphertext, how do we decrypt? Simply break the ciphertext into blocks of length n, change each to a vector, and multiply on the right by the inverse matrix N. In our example, we have

$$RBZ = (17, 1, 25) \mapsto (17, 1, 25)N = (755, 427, 66) \equiv (1, 11, 14) = blo,$$

and similarly for the remainder of the ciphertext.

For another example, see Example 21 in the Computer Appendices.

The Hill cipher is difficult to decrypt using only the ciphertext, but it succumbs easily to a known plaintext attack. If we do not know n, we can try various values until we find the right one. So suppose n is known. If we

have n of the blocks of plaintext of size n, then we can use the plaintext and the corresponding ciphertext to obtain a matrix equation for M (or for N, which might be more useful). For example, suppose we know that $n = 2$ and we have the plaintext

$$howareyoutoday =$$
$$7 \quad 14 \quad 22 \quad 0 \quad 17 \quad 4 \quad 24 \quad 14 \quad 20 \quad 19 \quad 14 \quad 3 \quad 0 \quad 24$$

corresponding to the ciphertext

$$ZWSENIUSPLJVEU =$$
$$25 \quad 22 \quad 18 \quad 4 \quad 13 \quad 8 \quad 20 \quad 18 \quad 15 \quad 11 \quad 9 \quad 21 \quad 4 \quad 20$$

The first two blocks yield the matrix equation

$$\begin{pmatrix} 7 & 14 \\ 22 & 0 \end{pmatrix} \begin{pmatrix} a & b \\ c & d \end{pmatrix} \equiv \begin{pmatrix} 25 & 22 \\ 18 & 4 \end{pmatrix} \pmod{26}.$$

Unfortunately, the matrix $\begin{pmatrix} 7 & 14 \\ 22 & 0 \end{pmatrix}$ has determinant -308, which is not invertible mod 26 (though this matrix could be used to reduce greatly the number of choices for the encryption matrix). Therefore, we replace the last row of the equation, for example, by the fifth block to obtain

$$\begin{pmatrix} 7 & 14 \\ 20 & 19 \end{pmatrix} \begin{pmatrix} a & b \\ c & d \end{pmatrix} \equiv \begin{pmatrix} 25 & 22 \\ 15 & 11 \end{pmatrix} \pmod{26}.$$

In this case, the matrix $\begin{pmatrix} 7 & 14 \\ 20 & 19 \end{pmatrix}$ is invertible mod 26:

$$\begin{pmatrix} 7 & 14 \\ 20 & 19 \end{pmatrix}^{-1} \equiv \begin{pmatrix} 5 & 10 \\ 18 & 21 \end{pmatrix} \pmod{26}.$$

We obtain

$$M \equiv \begin{pmatrix} 5 & 10 \\ 18 & 21 \end{pmatrix} \begin{pmatrix} 25 & 22 \\ 15 & 11 \end{pmatrix} \equiv \begin{pmatrix} 15 & 12 \\ 11 & 3 \end{pmatrix} \pmod{26}.$$

Because the Hill cipher is vulnerable to this attack, it cannot be regarded as being very strong.

A chosen plaintext attack proceeds by the same strategy, but is a little faster. Again, if you do not know n, try various possibilities until one works. So suppose n is known. Choose the first block of plaintext to be $baaa\cdots = 1000\ldots$, the second to be $abaa\cdots = 0100\ldots$, and continue through the nth block being $\ldots aaab = \ldots 0001$. The blocks of ciphertext will be the rows of the matrix M.

For a chosen ciphertext attack, use the same strategy as for chosen plaintext, where the choices now represent ciphertext. The resulting plaintext will be the rows of the inverse matrix N.

6.3 Modes of Operation

Suppose we have a block cipher. It can encrypt a block of plaintext of a fixed size, for example 64 bits. There are many circumstances, however, where it is necessary to encrypt messages that are either longer or shorter than the cipher's block length. For example, a bank may be sending a terabyte of data to another bank. Or you might be sending a short message that needs to be encrypted one letter at a time since you want to produce ciphertext output as quickly as you write the plaintext input.

Block ciphers can be run in many different modes of operation, allowing users to choose appropriate modes to meet the requirements of their applications. There are five common modes of operation: electronic codebook (ECB), cipher block chaining (CBC), cipher feedback (CFB), output feedback (OFB), and counter (CTR) modes. We now discuss these modes.

6.3.1 Electronic Codebook (ECB)

The natural manner for using a block cipher is to break a long piece of plaintext into appropriately sized blocks of plaintext and process each block separately with the encryption function E_K. This is known as the electronic codebook (ECB) mode of operation. The plaintext P is broken into smaller chunks $P = [P_1, P_2, \cdots, P_L]$ and the ciphertext is

$$C = [C_1, C_2, \cdots, C_L]$$

where $C_j = E_K(P_j)$ is the encryption of P_j using the key K.

There is a natural weakness in the ECB mode of operation that becomes apparent when dealing with long pieces of plaintext. Say an adversary Eve has been observing communication between Alice and Bob for a long enough period of time. If Eve has managed to acquire some plaintext pieces corresponding to the ciphertext pieces that she has observed, she can start to build up a codebook with which she can decipher future communication between Alice and Bob. Eve never needs to calculate the key K; she just looks up a ciphertext message in her codebook and uses the corresponding plaintext (if available) to decipher the message.

This can be a serious problem since many real-world messages consist of repeated fragments. E-mail is a prime example. An e-mail between Alice and Bob might start with the following header:

```
Date: Tue, 29 Feb 2000 13:44:38 -0500 (EST)
```

The ciphertext starts with the encrypted version of “Date: Tu”. If Eve finds that this piece of ciphertext often occurs on a Tuesday, she might be able to guess, without

knowing any of the plaintext, that such messages are e-mail sent on Tuesdays. With patience and ingenuity, Eve might be able to piece together enough of the message's header and trailer to figure out the context of the message. With even greater patience and computer memory, she might be able to piece together important pieces of the message.

Another problem that arises in ECB mode occurs when Eve tries to modify the encrypted message being sent to Bob. She might be able to extract important portions of the message and use her codebook to construct a false ciphertext message that she can insert in the data stream.

6.3.2 Cipher Block Chaining (CBC)

One method for reducing the problems that occur in ECB mode is to use chaining. Chaining is a feedback mechanism where the encryption of a block depends on the encryption of previous blocks.

In particular, encryption proceeds as

$$C_j = E_K(P_j \oplus C_{j-1}),$$

while decryption proceeds as

$$P_j = D_K(C_j) \oplus C_{j-1},$$

where C_0 is some chosen initial value. As usual, E_K and D_K denote the encryption and decryption functions for the block cipher.

Thus, in CBC mode, the plaintext is XORed with the previous ciphertext block and the result is encrypted. Figure 6.1 depicts CBC.

Figure 6.1 Cipher Block Chaining Mode.

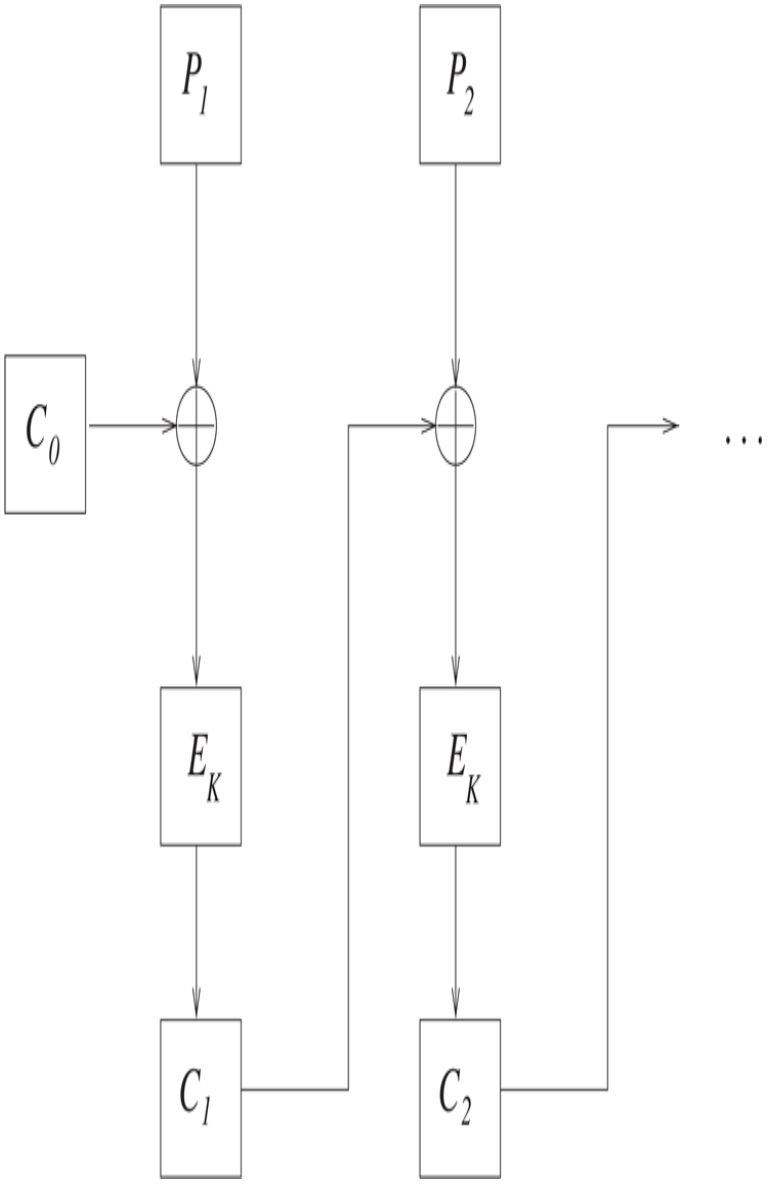

Figure 6.1 Full Alternative Text

The initial value C_0 is often called the initialization vector, or the IV. If we use a fixed value for C_0, say $C_0 = 0$, and ever have the same plaintext message, the result will be that the resulting ciphertexts will be the same. This is undesirable since it allows the adversary to deduce that the same plaintext was created. This can be

very valuable information, and can often be used by the adversary to infer the meaning of the original plaintext.

In practice, this problem is handled by always choosing C_0 randomly and sending C_0 in the clear along with the first ciphertext C_1. By doing so, even if the same plaintext message is sent repeatedly, an observer will see a different ciphertext each time.

6.3.3 Cipher Feedback (CFB)

One of the problems with both the CBC and ECB methods is that encryption (and hence decryption) cannot begin until a complete block of plaintext data is available. The cipher feedback mode operates in a manner that is very similar to the way in which LFSRs are used to encrypt plaintext. Rather than use linear recurrences to generate random bits, the cipher feedback mode is a stream mode of operation that produces pseudorandom bits using the block cipher E_K. In general, CFB operates in a k-bit mode, where each application produces k random bits for XORing with the plaintext. For our discussion, however, we focus on the eight-bit version of CFB. Using the eight-bit CFB allows one 8-bit piece of message (e.g., a single character) to be encrypted without having to wait for an entire block of data to be available. This is useful in interactive computer communications, for example.

For concreteness, let's assume that our block cipher encrypts blocks of 64 bits and outputs blocks of 64 bits (the sizes of the registers can easily be adjusted for other block sizes). The plaintext is broken into 8-bit pieces: $P = [P_1, P_2, \ldots]$, where each P_j has eight bits, rather than the 64 bits used in ECB and CBC. Encryption proceeds as follows. An initial 64-bit X_1 is chosen. Then for $j = 1, 2, 3, \ldots$, the following is performed:

$$
\begin{aligned}
O_j &= L_8(E_K(X_j)) \\
C_j &= P_j \oplus O_j \\
X_{j+1} &= R_{56}(X_j) \ \| \ C_j,
\end{aligned}
$$

where $L_8(X)$ denotes the 8 leftmost bits of X, $R_{56}(X)$ denotes the rightmost 56 bits of X, and $X \| Y$ denotes the string obtained by writing X followed by Y. We present the CFB mode of operation in Figure 6.2.

Figure 6.2 Cipher Feedback Mode.

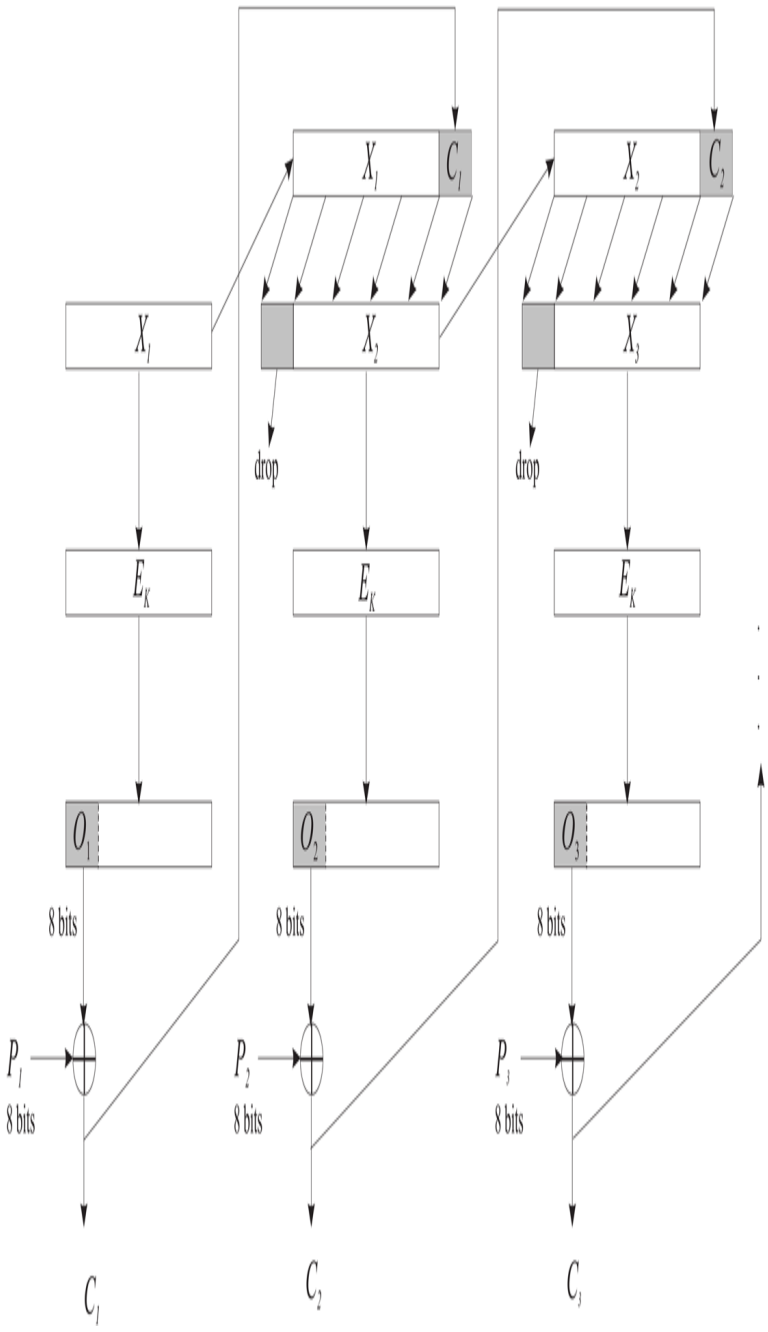

Figure 6.2 Full Alternative Text

Decryption is done with the following steps:

$$P_j = C_j \oplus L_8(E_K(X_j))$$
$$X_{j+1} = R_{56}(X_j) \ \| \ C_j.$$

We note that decryption does not involve the decryption function, D_K. This would be an advantage of running a block cipher in a stream mode in a case where the

decryption function for the block cipher is slower than the encryption function.

Let's step through one round of the CFB algorithm. First, we have a 64-bit register that is initialized with X_1. These 64 bits are encrypted using E_K and the leftmost eight bits of $E_K(X_1)$ are extracted and XORed with the 8-bit P_1 to form C_1. Then C_1 is sent to the recipient. Before working with P_2, the 64-bit register X_1 is updated by extracting the rightmost 56 bits. The eight bits of C_1 are appended on the right to form $X_2 = R_{56}(X_1) \parallel C_1$. Then P_2 is encrypted by the same process, but using X_2 in place of X_1. After P_2 is encrypted to C_2, the 64-bit register is updated to form

$$X_3 = R_{56}(X_2) \parallel C_2 = R_{48}(X_1) \parallel C_1 \parallel C_2.$$

By the end of the 8th round, the initial X_1 has disappeared from the 64-bit register and $X_9 = C_1 \parallel C_2 \parallel \cdots \parallel C_8$. The C_j continue to pass through the register, so for example $X_{20} = C_{12} \parallel C_{13} \parallel \cdots \parallel C_{19}$.

Note that CFB encrypts the plaintext in a manner similar to one-time pads or LFSRs. The key K and the numbers X_j are used to produce binary strings that are XORed with the plaintext to produce the ciphertext. This is a much different type of encryption than the ECB and CBC, where the ciphertext is the output of DES.

In practical applications, CFB is useful because it can recover from errors in transmission of the ciphertext. Suppose that the transmitter sends the ciphertext blocks $C_1, C_2, \ldots, C_k, \ldots$, and C_1 is corrupted during transmission, so that the receiver observes $\tilde{C}_1, C_2, \ldots$. Decryption takes $\tilde{C}_1$ and produces a garbled version of P_1 with bit errors in the locations that $\tilde{C}_1$ had bit errors. Now, after decrypting this ciphertext block, the receiver forms an incorrect X_2, which we denote $\tilde{X}_2$. If X_1 was $(*, *, *, *, *, *, *, *)$, then

$\tilde{X}_2 = (*, *, *, *, *, *, *, \tilde{C}_1)$. When the receiver gets an uncorrupted C_2 and decrypts, then a completely garbled version of P_2 is produced. When forming X_3, the decrypter actually forms $\tilde{X}_3 = (*, *, *, *, *, *, \tilde{C}_1, C_2)$. The decrypter repeats this process, ultimately getting bad versions of $P_1, P_2, \cdots, P_9$. When the decrypter calculates X_9, the error block has moved to the leftmost block of $\tilde{X}_9$ as $\tilde{X}_9 = (\tilde{C}_1, C_2, \cdots, C_8)$. At the next step, the error will have been flushed from the X_{10} register, and X_{10} and subsequent registers will be uncorrupted. For a simplified version of these ideas, see Exercise 18.

6.3.4 Output Feedback (OFB)

The CBC and CFB modes of operation exhibit a drawback in that errors propagate for a duration of time corresponding to the block size of the cipher. The output feedback mode (OFB) is another example of a stream mode of operation for a block cipher where encryption is performed by XORing the message with a pseudorandom bit stream generated by the block cipher. One advantage of the OFB mode is that it avoids error propagation.

Much like CFB, OFB may work on chunks of different sizes. For our discussion, we focus on the eight-bit version of OFB, where OFB is used to encrypt eight-bit chunks of plaintext in a streaming mode. Just as for CFB, we break our plaintext into eight-bit pieces, with $P = [P_1, P_2, \cdots,]$. We start with an initial value X_1, which has a length equal to the block length of the cipher, for example, 64 bits (the sizes of the registers can easily be adjusted for other block sizes). X_1 is often called the IV, and should be chosen to be random. X_1 is encrypted using the key K to produce 64 bits of output, and the leftmost eight bits O_1 of the ciphertext are extracted. These are then XORed with the first eight bits

P_1 of the plaintext to produce eight bits of ciphertext, C_1 .

So far, this is the same as what we were doing in CFB. But OFB differs from CFB in what happens next. In order to iterate, CFB updates the register X_2 by extracting the right 56 bits of X_1 and appending C_1 to the right side. Rather than use the ciphertext, OFB uses the output of the encryption. That is, OFB updates the register X_2 by extracting the right 56 bits of X_1 and appending O_1 to the right side.

In general, the following procedure is performed for $j = 1, 2, 3, \cdots$:

$$\begin{aligned} O_j &= L_8(E_K(X_j)) \\ X_{j+1} &= R_{56}(X_j) \ \| \ O_j \\ C_j &= P_j \oplus O_j. \end{aligned}$$

We depict the steps for the OFB mode of operation in Figure 6.3. Here, the output stream O_j is the encryption of the register containing the previous output from the block cipher. This output is then treated as a keystream and is XORed with the incoming plaintexts P_j to produce a stream of ciphertexts. Decryption is very simple. We get the plaintext P_j by XORing the corresponding ciphertext C_j with the output keystream O_j. Again, just like CFB, we do not need the decryption function D_K.

Figure 6.3 Output Feedback Mode.

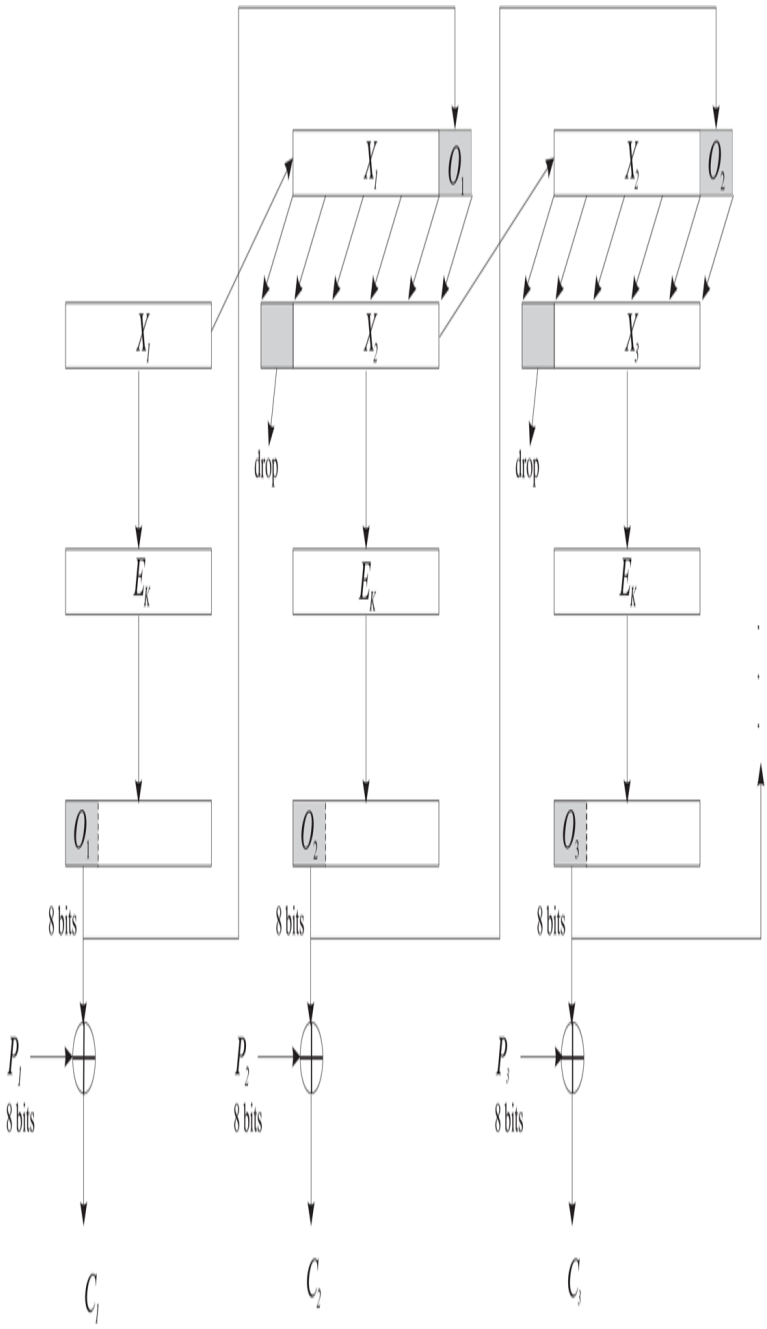

Figure 6.3 Full Alternative Text

So why would one want to build a stream cipher this way as opposed to the way the CFB stream cipher was built? There are a few key advantages to the OFB strategy. First, the generation of the O_j output key stream may be performed completely without any plaintext. What this means is that the key stream can be generated in advance. This might be desirable for applications where

we cannot afford to perform encryption operations as the plaintext message arrives.

Another advantage lies in its performance when errors are introduced to the ciphertext. Suppose a few errors are introduced to C_j when it is delivered to the receiver. Then only those corresponding bits in the plaintext are corrupted when decryption is performed. Since we build future output streams using the encryption of the register, and not using the corrupted ciphertext, the output stream will always remain clean and the errors in the ciphertext will not propagate.

To summarize, CFB required the register to completely flush itself of errors, which produced an entire block length of garbled plaintext bits. OFB, on the other hand, will immediately correct itself.

There is one problem associated with OFB, however, that is common to all stream ciphers that are obtained by XORing pseudorandom numbers with plaintext. If Eve knows a particular plaintext P_j and ciphertext C_j, she can conduct the following attack. She first calculates

$$O_j = C_j \oplus P_j$$

to get out the key stream. She may then create any false plaintext P_j' she wants. Now, to produce a ciphertext, she merely has to XOR with the output stream she calculated:

$$C_j' = P_j' \oplus O_j.$$

This allows her to modify messages.

6.3.5 Counter (CTR)

The counter (CTR) mode builds upon the ideas that were used in the OFB mode. Just like OFB, CTR creates an output key stream that is XORed with chunks of

plaintext to produce ciphertext. The main difference between CTR and OFB lies in the fact that the output stream O_j in CTR is not linked to previous output streams.

CTR starts with the plaintext broken into eight-bit pieces, $P = [P_1, P_2, \cdots]$. We begin with an initial value X_1, which has a length equal to the block length of the cipher, for example, 64 bits. Now, X_1 is encrypted using the key K to produce 64 bits of output, and the leftmost eight bits of the ciphertext are extracted and XORed with P_1 to produce eight bits of ciphertext, C_1.

Now, rather than update the register X_2 to contain the output of the block cipher, we simply take $X_2 = X_1 + 1$. In this way, X_2 does not depend on previous output. CTR then creates new output stream by encrypting X_2. Similarly, we may proceed by using $X_3 = X_2 + 1$, and so on. The jth ciphertext is produced by XORing the left eight bits from the encryption of the jth register with the corresponding plaintext P_j.

In general, the procedure for CTR is

$$
\begin{aligned}
X_j &= X_{j-1} + 1 \\
O_j &= L_8(E_K(X_j)) \\
C_j &= P_j \oplus O_j
\end{aligned}
$$

for $j = 2, 3, \cdots$, and is presented in Figure 6.4. The reader might wonder what happens to X_j if we continually add 1 to it. Shouldn't it eventually become too large? This is unlikely to happen, but if it does, we simply wrap around and start back at 0.

Figure 6.4 Counter Mode.

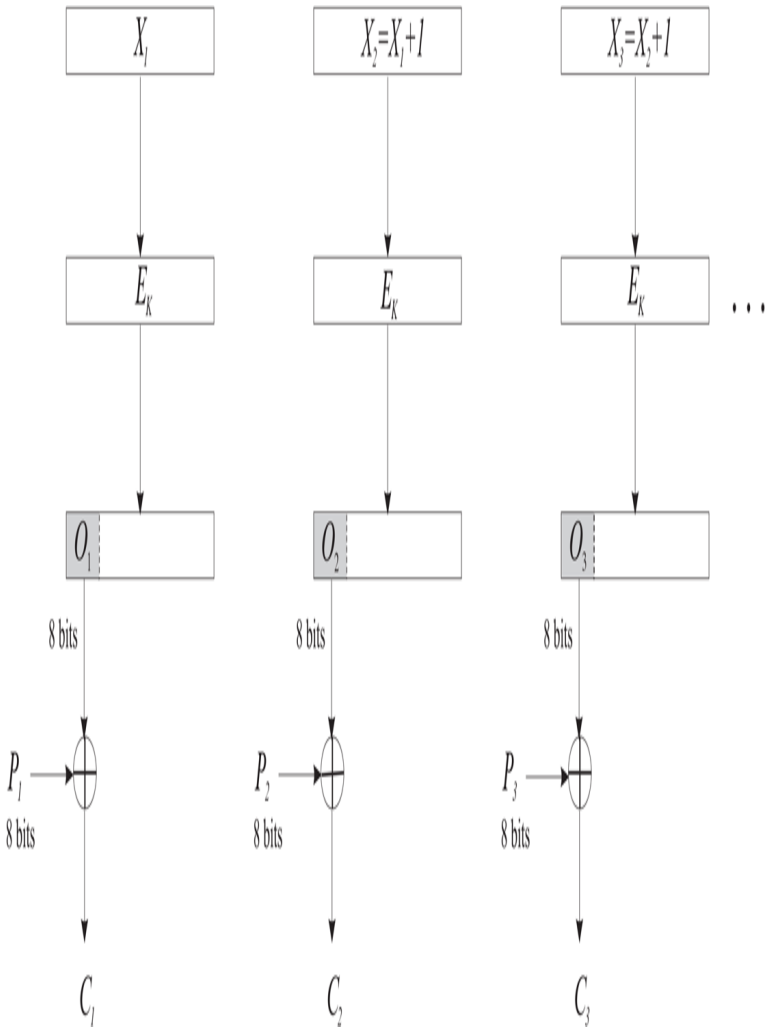

Figure 6.4 Full Alternative Text

Just like OFB, the registers X_j can be calculated ahead of time, and the actual encryption of plaintext is simple in that it involves just the XOR operation. As a result, its performance is identical to OFB's when errors are introduced in the ciphertext. The advantage of CTR mode compared to OFB, however, stems from the fact that many output chunks O_j may be calculated in parallel. We do not have to calculate O_j before calculating O_{j+1}. This makes CTR mode ideal for parallelizing.

6.4 Multiple Encryption

As technology improves and more sophisticated attacks are developed, encryptions systems become less secure and need to be replaced. There are two main approaches to achieving increased security. The first involves using encryption multiple times and leads, for example, to triple encryption. The second approach is to find a new system that is more secure, a potentially lengthy process.

We start by describing the idea behind multiple encryption schemes. The idea is to encrypt the same plaintext multiple times using the same algorithm with different keys. **Double encryption** encrypts the plaintext by first encrypting with one key and then encrypting again using another key. For example, if the keyspace for single encryption has 56 bits, hence 2^{56} keys, then the new keyspace consists of 2^{112} keys. One might guess that double encryption should therefore double the security. This, however, is not true. Merkle and Hellman showed that the double encryption scheme actually has the security level of a 57-bit key. The reduction from 2^{112} to 2^{57} makes use of the **meet-in-the-middle attack**, which is described in the next section.

Since double encryption has a weakness, **triple encryption** is often used. This appears to have a level of security approximately equivalent to a 112-bit key (when the single encryption has a 56-bit key). There are at least two ways that triple encryption can be implemented. One is to choose three keys, K_1, K_2, K_3, and perform $E_{K_1}(E_{K_2}(E_{K_3}(m)))$. This type of triple encryption is sometimes called *EEE*. The other is to choose two keys, K_1 and K_2, and perform $E_{K_1}(D_{K_2}(E_{K_1}(m)))$. This is sometimes called *EDE*. When $K_1 = K_2$, this reduces to

single encryption. Therefore, a triple encryption machine that is communicating with an older machine that still uses single encryption can simply set $K_1 = K_2$ and proceed. This compatibility is the reason for using D_{K_2} instead of E_{K_2} in the middle; the use of D instead of E gives no extra cryptographic strength. Both versions of triple encryption are resistant to meet-in-the-middle attacks (compare with Exercise 11). However, there are other attacks on the two-key version ([Merkle-Hellman] and [van Oorschot-Wiener]) that indicate possible weaknesses, though they require so much memory as to be impractical.

Another strengthening of encryption was proposed by Rivest. Choose three keys, K_1, K_2, K_3, and perform $K_3 \oplus E_{K_2}(K_1 \oplus m)$. In other words, modify the plaintext by *XOR*ing with K_1, then apply encryption with K_2, then *XOR* the result with K_3. This method, when used with DES, is known as DESX and has been shown to be fairly secure. See [Kilian-Rogaway].

6.5 Meet-in-the-Middle Attacks

Alice and Bob are using an encryption method. The encryption functions are called E_K, and the decryption functions are called D_K, where K is a key. We assume that if someone knows K, then she also knows E_K and D_K (so Alice and Bob could be using one of the classical, nonpublic key systems such as DES or AES). They have a great idea. Instead of encrypting once, they use two keys K_1 and K_2 and encrypt twice. Starting with a plaintext message m, the ciphertext is $c = E_{K_2}(E_{K_1}(m))$. To decrypt, simply compute $m = D_{K_1}(D_{K_2}(c))$. Eve will need to discover both K_1 and K_2 to decrypt their messages.

Does this provide greater security? For many cryptosystems, applying two encryptions is the same as using an encryption for some other key. For example, the composition of two affine functions is still an affine function (see Exercise 11 in Chapter 2). Similarly, using two RSA encryptions (with the same n) with exponents e_1 and e_2 corresponds to doing a single encryption with exponent e_1e_2. In these cases, double encryption offers no advantage. However, there are systems, such as DES (see Subsection 7.4.1) where the composition of two encryptions is not simply encryption with another key. For these, double encryption might seem to offer a much higher level of security. However, the following attack shows that this is not really the case, as long as we have a computer with a lot of memory.

Assume Eve has intercepted a message m and a doubly encrypted ciphertext $c = E_{K_2}(E_{K_1}(m))$. She wants to find K_1 and K_2. She first computes two lists:

1. $E_K(m)$ for all possible keys K
2. $D_L(c)$ for all possible keys L.

Finally, she compares the two lists and looks for matches. There will be at least one match, since the correct pair of keys will be one of them, but it is likely that there will be many matches. If there are several matches, she then takes another plaintext–ciphertext pair and determines which of the pairs (K, L) she has found will encrypt the plaintext to the ciphertext. This should greatly reduce the list. If there is still more than one pair remaining, she continues until only one pair remains (or she decides that two or more pairs give the same double encryption function). Eve now has the desired pair K_1, K_2.

If Eve has only one plaintext–ciphertext pair, she still has reduced the set of possible key pairs to a short list. If she intercepts a future transmission, she can try each of these possibilities and obtain a very short list of meaningful plaintexts.

If there are N possible keys, Eve needs to compute N values $E_L(m)$. She then needs to compute N numbers $D_L(c)$ and compare them with the stored list. But these $2N$ computations (plus the comparisons) are much less than the N^2 computations required for searching through all key pairs K_1, K_2.

This meet-in-the-middle procedure takes slightly longer than the exhaustive search through all keys for single encryption. It also takes a lot of memory to store the first list. However, the conclusion is that double encryption does not significantly raise the level of security.

Similarly, we could use triple encryption, using triples of keys. A similar attack brings the level of security down to at most what one might naively expect from double

encryption, namely squaring the possible number of keys.

Example

Suppose the single encryption has 2^{56} possible keys and the block cipher inputs and outputs blocks of 64 bits, as is the case with DES. The first list has 2^{56} entries, each of which is a 64-bit block. The probability that a given block in List 1 matches a given block in List 2 is 2^{-64}. Since there are 2^{56} entries in List 2, we expect that a given block in List 1 matches $2^{56}2^{-64} = 2^{-8}$ entries of List 2. Running through the 2^{56} elements in List 1, we expect $2^{56}2^{-8} = 2^{48}$ pairs (K, L) for which there are matches between List 1 and List 2.

We know that one of these 2^{48} matches is from the correct pair (K_1, K_2), and the other matches are probably caused by randomness. If we take a random pair (K, L) and try it on a new plaintext–ciphertext (m_1, c_1), then $E_L(E_K(m_1))$ is a 64-bit block that has probability 2^{-64} of matching the 64-bit block c_1. Therefore, among the approximately 2^{48} random pairs, we expect $2^{48}2^{-64} = 2^{-16}$ matches between $E_L(E_K(m_1))$ and c_1. In other words, it is likely that the second plaintext–ciphertext pair eliminates all extraneous solutions and leaves only the correct key pair (K_1, K_2). If not, a third round should complete the task.

6.6 Exercises

1. The ciphertext *YIFZMA* was encrypted by a Hill cipher with matrix $\begin{pmatrix} 9 & 13 \\ 2 & 3 \end{pmatrix}$. Find the plaintext.

2. The matrix $\begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}$ mod 26 is not suitable for the matrix in a Hill cipher. Why?

3. The ciphertext text *GEZXDS* was encrypted by a Hill cipher with a 2×2 matrix. The plaintext is *solved*. Find the encryption matrix M.

4. Consider the following combination of Hill and Vigenère ciphers: The key consists of three 2×2 matrices, M_1, M_2, M_3. The plaintext letters are represented as integers mod 26. The first two are encrypted by M_1, the next two by M_2, the 5th and 6th by M_3. This is repeated cyclically, as in the Vigenère cipher. Explain how to do a chosen plaintext attack on this system. Assume that you know that three 2×2 matrices are being used. State explicitly what plaintexts you would use and how you would use the outputs.

5. Eve captures Bob's Hill cipher machine, which uses a 2-by-2 matrix M mod 26. She tries a chosen plaintext attack. She finds that the plaintext *ba* encrypts to HC and the plaintext *zz* encrypts to GT. What is the matrix M.

6. Alice uses a Hill cipher with a 3×3 matrix M that is invertible mod 26. Describe a chosen plaintext attack that will yield the entries of the matrix M. Explicitly say what plaintexts you will use.

7.
    1. The ciphertext text *ELNI* was encrypted by a Hill cipher with a 2×2 matrix. The plaintext is *dont*. Find the encryption matrix.

    2. Suppose the ciphertext is *ELNK* and the plaintext is still *dont*. Find the encryption matrix. Note that the second column of the matrix is changed. This shows that the entire second column of the encryption matrix is involved in obtaining the last character of the ciphertext.

8. Suppose the matrix $\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$ is used for an encryption matrix in a Hill cipher. Find two plaintexts that encrypt to the same ciphertext.

9. Let a, b, c, d, e, f be integers mod 26. Consider the following combination of the Hill and affine ciphers: Represent a block of plaintext as a pair (x, y) mod 26. The corresponding ciphertext (u, v) is

$$(x \quad y)\begin{pmatrix} a & b \\ c & d \end{pmatrix} + (e \quad f) \equiv (u \quad v) \pmod{26}.$$

   Describe how to carry out a chosen plaintext attack on this system (with the goal of finding the key a, b, c, d, e, f). You should state explicitly what plaintexts you choose and how to recover the key.

10. Alice is sending a message to Bob using a Hill cipher with a 2×2 matrix. In fact, Alice is bored and her plaintext consists of the letter a repeated a few hundred times. Eve knows what system is being used, but not the key, and intercepts the ciphertext. State how Eve will recognize that the plaintext is one repeated letter and decide whether or not Eve can deduce the letter and/or the key. (Note: The solution very much depends on the fact that the repeated letter is a, rather than b, c, ...)

11. Let E_K denote encryption (for some cryptosystem) with key K. Suppose that there are 2^{35} possible keys K. Alice decides to encrypt a message m as follows:

    She chooses two keys K and L and double encrypts by computing

$$c = E_K(E_K(E_L(m)))$$

    to get the ciphertext c. Suppose Eve knows Alice's method of encryption (but not K and L) and has at least two plaintext–ciphertext pairs. Describe a method that is *guaranteed* to yield the correct K and L (and maybe a *very small* additional set of incorrect pairs). Be explicit enough to say why you are using at least two plaintext–ciphertext pairs. Eve may do up to 2^{50} computations.

12. Alice and Bob are arguing about which method of multiple encryption they should use. Alice wants to choose keys K_1 and K_2 and triple encrypt a message m as $c = E_{K_1}(E_{K_2}(E_{K_1}(m)))$. Bob wants to encrypt m as $c = E_{K_1}(E_{K_1}(E_{K_2}(E_{K_2}(m))))$. Which method is more secure? Describe in detail an attack on the weaker encryption method.

13. Alice and Bob are trying to implement triple encryption. Let E_K denote DES encryption with key K and let D_K denote decryption.

    1. Alice chooses two keys, K_1 and K_2, and encrypts using the formula $c = E_{K_1}(D_{K_2}(E_{K_1}(m)))$. Bob chooses two keys, K_1 and K_2, and encrypts using the formula $c = E_{K_1}(E_{K_2}(E_{K_2}(m)))$. One of these methods is more secure than the other. Say which one is weaker and explicitly give the steps that can be used to attack the

weaker system. You may assume that you know ten plaintext–ciphertext pairs.

2. What is the advantage of using D_{K_2} instead of E_{K_2} in Alice's version?

14. Suppose E^1 and E^2 are two encryption methods. Let K_1 and K_2 be keys and consider the double encryption

$$E_{K_1,\,K_2}(m) = E^1_{K_1}(E^2_{K_2}(m)).$$

1. Suppose you know a plaintext–ciphertext pair. Show how to perform a meet-in-the-middle attack on this double encryption.

2. An affine encryption given by $x \mapsto \alpha x + \beta \pmod{26}$ can be regarded as a double encryption, where one encryption is multiplying the plaintext by α and the other is a shift by β. Assume that you have a plaintext and ciphertext that are long enough that α and β are unique. Show that the meet-in-the-middle attack from part (a) takes at most 38 steps (not including the comparisons between the lists). Note that this is much faster than a brute force search through all 312 keys.

15. Let E_K denote DES encryption with key K. Suppose there is a public database Y consisting of 10^{10} DES keys and there is another public database Z of 10^{10} binary strings of length 64. Alice has five messages $m_1, m_2, \ldots, m_5$. She chooses a key K from Y and a string B from Z. She encrypts each message m by computing $c = E_K(m) \oplus B$. She uses the same K and B for each of the messages. She shows the five plaintext–ciphertext pairs (m_i, c_i) to Eve and challenges Eve to find K and B. Alice knows that Eve's computer can do only 10^{15} calculations, and there are 10^{20} pairs (K, B), so Alice thinks that Eve cannot find the correct pair. However, Eve has taken a crypto course. Show how she can find the K and B that Alice used. You must state explicitly what Eve does. Statements such as "Eve makes a list" are not sufficient; you must include what is on the lists and how long they are.

16. Alice wants to encrypt her messages securely, but she can afford only an encryption machine that uses a 25-bit key. To increase security, she chooses 4 keys K_1, K_2, K_3, K_4 and encrypts four times:

$$c = E_{K_1}(E_{K_2}(E_{K_3}(E_{K_4}(m)))).$$

Eve finds several plaintext–ciphertext pairs (m, c) encrypted with this set of keys. Describe how she can find (with high probability) the keys K_1, K_2, K_3, K_4. (For this problem, assume that Eve can do at most 2^{60} computations, so she cannot try all 2^{100} combinations of keys.) (Note: If you use only one of the plaintext–

ciphertext pairs in your solution, you probably have not done enough to determine the keys.)

17. Show that the decryption procedures given for the CBC and CFB modes actually perform the desired decryptions.

18. Consider the following simplified version of the CFB mode. The plaintext is broken into 32-bit pieces: $P = [P_1, P_2, \ldots]$, where each P_j has 32 bits, rather than the eight bits used in CFB. Encryption proceeds as follows. An initial 64-bit X_1 is chosen. Then for $j = 1, 2, 3, \ldots$, the following is performed:

$$C_j = P_j \oplus L_{32}(E_K(X_j))$$
$$X_{j+1} = R_{32}(X_j) \parallel C_j,$$

where $L_{32}(X)$ denotes the 32 leftmost bits of X, $R_{32}(X)$ denotes the rightmost 32 bits of X, and $X \parallel Y$ denotes the string obtained by writing X followed by Y.

    1. Find the decryption algorithm.

    2. The ciphertext consists of 32-bit blocks $C_1, C_2, C_3, C_4, \ldots$. Suppose that a transmission error causes C_1 to be received as $\tilde{C}_1 \neq C_1$, but that $C_2, C_3, C_4, \ldots$ are received correctly. This corrupted ciphertext is then decrypted to yield plaintext blocks $\tilde{P}_1, \tilde{P}_2, \ldots$. Show that $\tilde{P}_1 \neq P_1$, but that $\tilde{P}_i = P_i$ for all $i \geq 4$. Therefore, the error affects only three blocks of the decryption.

19. The cipher block chaining (CBC) mode has the property that it recovers from errors in ciphertext blocks. Show that if an error occurs in the transmission of a block C_j, but all the other blocks are transmitted correctly, then this affects only two blocks of the decryption. Which two blocks?

20. In CTR mode, the initial X_1 has 64 bits and is sent unencrypted to the receiver. (a) If X_1 is chosen randomly every time a message is encrypted, approximately how many messages must be sent in order for there to be a good chance that two messages use the same X_1 ? (b) What could go wrong if the same X_1 is used for two different messages? (Assume that the key K is not changed.)

21. Suppose that in CBC mode, the final plaintext block P_n is incomplete; that is, its length M is less than the usual block size of, say, 64 bits. Often, this last block is padded with a binary string to make it have full length. Another method that can be used is called **ciphertext stealing**, as follows:

    1. Compute $Y_{n-1} = E_K(C_{n-2} \oplus P_{n-1})$.

    2. Compute $C_n = L_M(Y_{n-1})$, where L_M means we take the leftmost M bits.

3. Compute $C_{n-1} = E_K\big((P_n||0^{64-M}) \oplus Y_{n-1}\big)$, where $P_n||0^{64-M}$ denotes P_n with enough 0s appended to give it the length of a full 64-bit block.

4. The ciphertext is $C_1C_2 \cdots C_{n-1}C_n$. Therefore, the ciphertext has the same length as the plaintext.

Suppose you receive a message that used this ciphertext stealing for the final blocks (the ciphertext blocks $C_1, \ \ldots C_{n-2}$ were computed in the usual way for CBC). Show how to decrypt the ciphertext (you have the same key as the sender).

22. Suppose Alice has a block cipher with 2^{50} keys, Bob has one with 2^{40} keys, and Carla has one with 2^{30} keys. The only known way to break single encryption with each system is by brute force, namely trying all keys. Alice uses her system with single encryption. But Bob uses his with double encryption, and Carla uses hers with triple encryption. Who has the most secure system? Who has the weakest? (Assume that double and triple encryption do not reduce to using single or double encryption, respectively. Also, assume that some plaintext-ciphertext pairs are available for Alice's single encryption, Bob's double encryption, and Carla's triple encryption.)

6.7 Computer Problems

1. The following is the ciphertext of a Hill cipher

   ```
   zirkzwopjjoptfapuhfhadrq
   ```

   using the matrix

   Decrypt.