4.2. Encrypting a Large Message

How do you encrypt a message larger than 64 bits? There are several schemes defined in [DES81]. These schemes would be equally applicable to IDEA or any secret key scheme that encrypted fixed-length blocks, and no doubt one could come up with variant schemes as well. The ones defined in [DES81], and which we'll describe in detail, are:

1. Electronic Code Book (ECB)
2. Cipher Block Chaining (CBC)
3. *k*-Bit Cipher Feedback Mode (CFB)
4. *k*-Bit Output Feedback Mode (OFB)

   A newer scheme that might be important in the future is:

5. Counter Mode (CTR)

4.2.1. Electronic Code Book (ECB)

This mode consists of doing the obvious thing, and it is usually the worst method. You break the message into 64-bit blocks (padding the last one out to a full 64 bits), and encrypt each block with the secret key (see Figure 4-1). The other side receives the encrypted blocks and decrypts each block in turn to get back the original message (see Figure 4-2).

Figure 4-1. Electronic Code Book Encryption

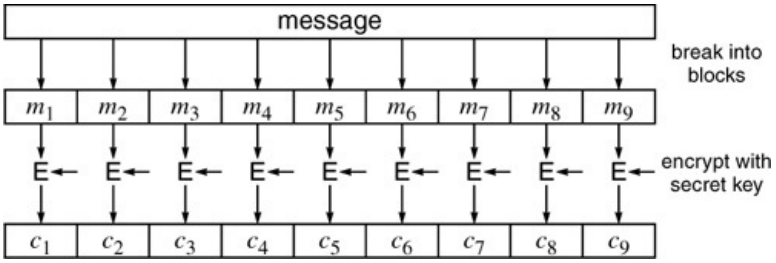
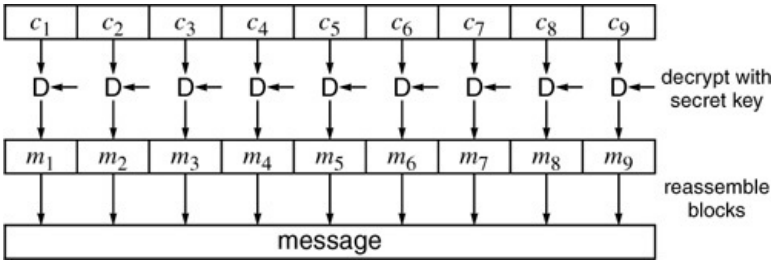


Figure 4-2. Electronic Code Book Decryption



There are a number of problems with this approach that don't show up in the single-block case. First, if a message contains two identical 64-bit blocks, the corresponding two blocks of

ciphertext will be identical. This will give an eavesdropper some information. Whether it is useful or not depends on the context. We'll give an example where ECB would have a problem. Suppose that the eavesdropper knows that the plaintext is an alphabetically sorted list of employees and salaries being sent from management to payroll, tabularly arranged (see Figure 4-3).

Figure 4-3. Payroll Data

Name	Position	Salary
Adams, John	President	78,964.31
Bush, Neil	Accounting Clerk	623,321.16
Hoover, J. Edgar	Wardrobe Consultant	34,445.22
Stern, Howard	Affirmative Action Officer	38,206.51
Woods, Rosemary	Audiovisual Supervisor	21,489.15

Block boundaries

Further suppose that, as luck would have it, each line is exactly 64 bytes long, and the blocks happen to be divided in the salary field between the 1,000's and the 10,000's digit. Since identical plaintext blocks produce identical ciphertext blocks, not only can an eavesdropper figure out which sets of employees have identical salaries, but also which sets of employees have salaries in the same $10,000 ranges. If he can guess a few relative salaries, he will have a pretty good idea of what all the salaries are from this "encrypted" message.

Furthermore, if the eavesdropper is one of the employees, he can alter the message to change his own salary to match that of any other employee by copying the ciphertext blocks from that employee to the corresponding blocks of his own entry. Even a human looking at the resulting message would see nothing awry.

So, ECB has two serious flaws. Someone seeing the ciphertext can gain information from repeated blocks, and someone can rearrange blocks or modify blocks to his own advantage. As a result of these flaws, ECB is rarely used to encrypt messages.

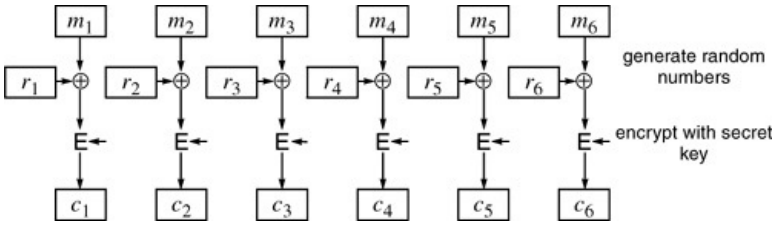
4.2.2. Cipher Block Chaining (CBC)

CBC is a method of avoiding some of the problems in ECB. Using CBC, even if the same block repeats in the plaintext, it will not cause repeats in the ciphertext.

First we'll give an example of how this might be accomplished. Our example is not CBC, but it helps for understanding CBC.

Generate a 64-bit random number r_i for each plaintext block m_i to be encrypted. $\oplus$ the plaintext block with the random number, encrypt the result, and transmit both the unencrypted random number r_i and the ciphertext block c_i (see Figure 4-4). To decrypt this, you'd decrypt all the c_is, and for each c_i, after decrypting it, you'd $\oplus$ it with the random number r_i.

Figure 4-4. Randomized Electronic Code Book Encryption

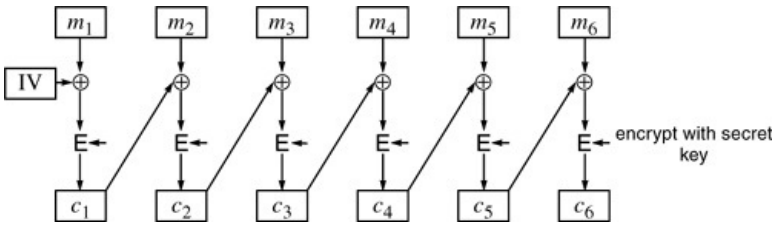


The main problem with this scheme is efficiency. It causes twice as much information to be

transmitted, since a random number has to be transmitted along with each block of ciphertext. Another problem with it is that an attacker can rearrange the blocks and have a predictable effect on the resulting plaintext. For instance, if $r_2|c_2$ were removed entirely it would result in m_2 being absent in the decrypted plaintext. Or if $r_2|c_2$ were swapped with $r_7|c_7$, then m_2 and m_7 would be swapped in the result. Worse yet, an attacker knowing the value of any block m_n can change it in a predictable way by making the corresponding change in r_n.

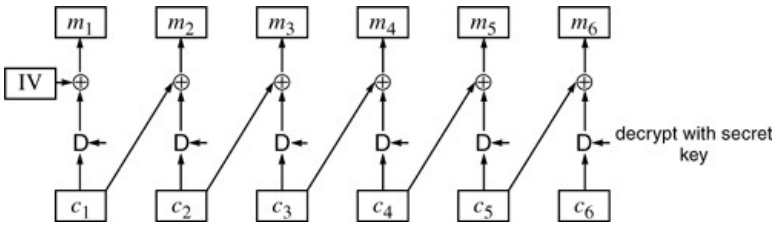
Now we can explain CBC. CBC generates its own "random numbers". It uses c_i as r_{i+1}. In other words, it takes the previous block of ciphertext and uses that as the random number that will be $\oplus$'d into the next plaintext. To avoid having two plaintext messages that start the same wind up with the same ciphertext in the beginning, CBC does select one random number, which gets $\oplus$'d into the first block of plaintext, and transmits it along with the data. This initial random number is known as an **IV** (**initialization vector**).

Figure 4-5. Cipher Block Chaining Encryption



Decryption is simple because $\oplus$ is its own inverse.

Figure 4-6. Cipher Block Chaining Decryption



Since the cost of the $\oplus$ is trivial compared to the cost of an encryption, CBC encryption has the same performance as ECB encryption except for the cost of generating and transmitting the IV.

In many cases the security of CBC would not be adversely affected by omitting the IV (or, equivalently, using the value 0 as the IV). However, we'll give one example where it would matter. Suppose the encrypted file of employees and salaries is transmitted weekly. If there were no IV, then an eavesdropper could tell where the ciphertext first differed from the previous week, and therefore perhaps determine the first person whose salary had changed.

Another example is where a general sends information each day saying continue holding your position. The ciphertext will be the same every day, until the general decides to send something else, like start bombing. Then the ciphertext would suddenly change, alerting the enemy.

A randomly chosen IV guarantees that even if the same message is sent repeatedly, the ciphertext will be completely different each time.

Finally, a randomly chosen IV prevents attackers from supplying chosen plaintext to the underlying encryption algorithm even if they can supply chosen plaintext to the CBC.