Chapter 5
More About Block Ciphers

A block cipher is much more than just an encryption algorithm. It can be used as a versatile building block with which a diverse set of cryptographic mechanisms can be realized. For instance, we can use them for building different types of block-based encryption schemes, and we can even use block ciphers for realizing stream ciphers. The different ways of encryption are called *modes of operation* and are discussed in this chapter. Block ciphers can also be used for constructing hash functions, message authentication codes which are also knowns as MACs, or key establishment protocols, all of which will be described in later chapters. There are also other uses for block ciphers, e.g., as pseudo-random generators. In addition to modes of operation, this chapter also discusses two very useful techniques for increasing the security of block ciphers, namely key whitening and multiple encryption.

In this chapter you will learn

- the most important modes of operation for block ciphers in practice
- security pitfalls when using modes of operations
- the principles of key whitening
- why double encryption is not a good idea, and the meet-in-the-middle attack
- triple encryption

5.1 Encryption with Block Ciphers: Modes of Operation

In the previous chapters we introduced how DES, 3DES and AES encrypt a block of data. Of course, in practice one wants typically to encrypt more than one single 8-byte or 16-byte block of plaintext, e.g., when encrypting an e-mail or a computer file. There are several ways of encrypting long plaintexts with a block cipher. We introduce several popular modes of operation in this chapter, including

- Electronic Code Book mode (ECB),
- Cipher Block Chaining mode (CBC),
- Cipher Feedback mode (CFB),
- Output Feedback mode (OFB),
- Counter mode (CTR).

The latter three modes use the block cipher as a building block for a stream cipher.

All of the five modes have one goal: They encrypt data and thus provide confidentiality for a message sent from Alice to Bob. In practice, we often not only want to keep data confidential, but Bob also wants to know whether the message is really coming from Alice. This is called authentication and the Galois Counter mode (GCM), which we will also introduce, is a mode of operation that lets the receiver (Bob) determine whether the message was really sent by the person he shares a key with (Alice). Moreover, authentication also allows Bob to detect whether the ciphertext was altered during transmission. More on authentication is found in Chap. 10.

The ECB and CFB modes require that the length of the plaintext be an exact multiple of the block size of the cipher used, e.g., a multiple of 16 bytes in the case of AES. If the plaintext does not have this length, it must be padded. There are several ways of doing this padding in practice. One possible padding method is to append a single "1" bit to the plaintext and then to append as many "0" bits as necessary to reach a multiple of the block length. Should the plaintext be an exact multiple of the block length, an extra block consisting only of padding bits is appended.

5.1.1 Electronic Codebook Mode (ECB)

The *Electronic Code Book (ECB)* mode is the most straightforward way of encrypting a message. In the following, let $e_k(x_i)$ denote the encryption of plaintext block x_i with key k using some arbitrary block cipher. Let $e_k^{-1}(y_i)$ denote the decryption of ciphertext block y_i with key k. Let us assume that the block cipher encrypts (decrypts) blocks of size b bits. Messages which exceed b bits are partitioned into b-bit blocks. If the length of the message is not a multiple of b bits, it must be padded to a multiple of b bits prior to encryption. As shown in Fig. 5.1, in ECB mode each block is encrypted separately. The block cipher can, for instance, be AES or 3DES.

Encryption and decryption in the ECB mode is formally described as follows:

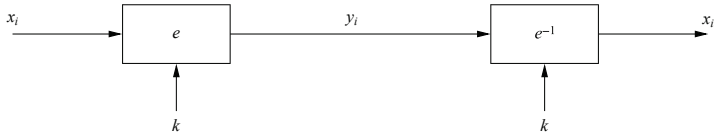

Fig. 5.1 Encryption and decryption in ECB mode

Definition 5.1.1 Electronic Codebook Mode (ECB)
Let $e()$ be a block cipher of block size b, and let x_i and y_i be bit strings of length b.
***Encryption**: $y_i = e_k(x_i), \quad i \geq 1$*
***Decryption**: $x_i = e_k^{-1}(y_i) = e_k^{-1}(e_k(x_i)), \quad i \geq 1$*

It is straightforward to verify the correctness of the ECB mode:

$$e_k^{-1}(y_i) = e_k^{-1}(e_k(x_i)) = x_i.$$

The ECB mode has advantages. Block synchronization between the encryption and decryption parties Alice and Bob is not necessary, i.e., if the receiver does not receive all encrypted blocks due to transmission problems, it is still possible to decrypt the received blocks. Similarly, bit errors, e.g., caused by noisy transmission lines, only affect the corresponding block but not succeeding blocks. Also, block ciphers operating in ECB mode can be parallelized, e.g., one encryption unit encrypts (or decrypts) block 1, the next one block 2, and so on. This is an advantage for high-speed implementations, but many other modes such as the CFB do not allow parallelization.

However, as is often the case in cryptography, there are some unexpected weaknesses associated with the ECB mode which we will discuss in the following. The main problem of the ECB mode is that it encrypts highly deterministically. This means that identical plaintext blocks result in identical ciphertext blocks, as long as the key does not change. The ECB mode can be viewed as a gigantic code book — hence the mode's name — which maps every input to a certain output. Of course, if the key is changed the entire code book changes, but as long as the key is static the book is fixed. This has several undesirable consequences. First, an attacker recognizes if the same message has been sent twice simply by looking at the ciphertext. Deducing information from the ciphertext in this way is called *traffic analysis*. For instance, if there is a fixed header that always precedes a message, the header always results in the same ciphertext. From this, an attacker can, for instance, learn when a new message has been sent. Second, plaintext blocks are encrypted independently of previous blocks. If an attacker reorders the ciphertext blocks, this might result in valid plaintext and the reordering might not be detected. We demonstrate two simple attacks which exploit these weaknesses of the ECB mode.

The ECB mode is susceptible to *substitution attacks*, because once a particular plaintext to ciphertext block mapping $x_i \rightarrow y_i$ is known, a sequence of ciphertext

blocks can easily be manipulated. We demonstrate how a substitution attack could work in the real world. Imagine the following example of an electronic wire transfer betweens banks.

Example 5.1. Substitution attack against electronic bank transfer
Let's assume a protocol for wire transfers between banks (Fig. 5.2). There are five fields which specify a transfer: the sending bank's ID and account number, the receiving bank's ID and account number, and the amount. We assume now (and this is a major simplification) that each of the fields has exactly the size of the block cipher width, e.g., 16 bytes in the case of AES. Furthermore, the encryption key between the two banks does not change too frequently. Due to the nature of the ECB, an attacker can exploit the deterministic nature of this mode of operation by simple substitution of the blocks. The attack details are as follows:

Block #	1	2	3	4	5
	Sending Bank A	Sending Account #	Receiving Bank B	Receiving Account #	Amount $

Fig. 5.2 Example for a substitution attack against ECB encryption

1. The attacker, Oscar, opens one account at bank A and one at bank B.
2. Oscar taps the encrypted line of the banking communication network.
3. He sends $1.00 transfers from his account at bank A to his account at bank B repeatedly. He observes the ciphertexts going through the communication network. Even though he cannot decipher the random-looking ciphertext blocks, he can check for ciphertext blocks that repeat. After a while he can recognize the five blocks of his own transfer. He now stores blocks 1, 3 and 4 of these transfers. These are the encrypted versions of the ID numbers of both banks as well as the encrypted version of his account at bank B.
4. Recall that the two banks do not change the key too frequently. This means that the same key is used for several other transfers between bank A and B. By comparing blocks 1 and 3 of *all* subsequent messages with the ones he has stored, Oscar recognizes all transfers that are made from some account at bank A to some account at bank B. He now simply replaces block 4 — which contains the receiving account number — with the block 4 that he stored before. This block contains Oscar's account number in encrypted form. As a consequence, *all transfers* from some account of bank A to some account of bank B are redirected to go into Oscar's B account! Note that bank B now has means of detecting that the block 4 has been replaced in some of the transfers it receives.
5. Withdraw money from bank B quickly and fly to a country that has a relaxed attitude about the extradition of white-collar criminals.

◇

What's interesting about this attack is that it works completely without attacking the block cipher itself. So even if we would use AES with a 256-bit key and if