Cryptography: An Introduction (3rd Edition)

Nigel Smart

Chapter 8 — Edited for CIS 331 (several content cuts)

CHAPTER 8

Block Ciphers

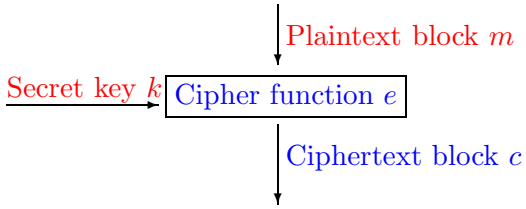
Chapter Goals

- To introduce the notion of block ciphers.
- To understand the workings of the DES algorithm.
- To understand the workings of the Rijndael algorithm.
- To learn about the various standard modes of operation of block ciphers.

1. Introduction To Block Ciphers

The basic description of a block cipher is shown in Fig. 1. Block ciphers operate on blocks

FIGURE 1. Operation of a block cipher



of plaintext one at a time to produce blocks of ciphertext. The main difference between a block cipher and a stream cipher is that block ciphers are stateless, whilst stream ciphers maintain an internal state which is needed to determine which part of the keystream should be generated next. We write

$$c = e_k(m),$$
$$m = d_k(c)$$

where

- m is the plaintext block,
- k is the secret key,
- e is the encryption function,
- d is the decryption function,
- c is the ciphertext block.

The block sizes taken are usually reasonably large, 64 bits in DES and 128 bits or more in modern block ciphers. Often the output of the ciphertext produced by encrypting the first block is used to help encrypt the second block in what is called a *mode of operation*. These modes are used to avoid certain attacks based on deletion or insertion by giving each ciphertext block a context within the overall message. Each mode of operation offers different protection against error propagation due to transmission errors in the ciphertext. In addition, depending on the mode of operation (and the application) message/session keys may be needed. For example, many modes require a per message

initial value to be input into the encryption and decryption operations. Later in this chapter we shall discuss modes of operation of block ciphers in more detail.

There are many block ciphers in use today, some which you may find used in your web browser are RC5, RC6, DES or 3DES. The most famous of these is DES, or the Data Encryption Standard. This was first published in the mid-1970s as a US Federal standard and soon become the de-facto international standard for banking applications.

The DES algorithm has stood up remarkably well to the test of time, but in the early 1990s it became clear that a new standard was required. This was because both the block length (64 bits) and the key length (56 bits) of basic DES were too small for future applications. It is now possible to recover a 56-bit DES key using either a network of computers or specialized hardware. In response to this problem the US National Institute for Standards and Technology (NIST) initiated a competition to find a new block cipher, to be called the Advanced Encryption Standard or AES.

Unlike the process used to design DES, which was kept essentially secret, the design of the AES was performed in public. A number of groups from around the world submitted designs for the AES. Eventually five algorithms, known as the AES finalists, were chosen to be studied in depth. These were

- MARS from a group at IBM,
- RC6 from a group at RSA Security,
- Twofish from a group based at Counterpane, UC Berkeley and elsewhere,
- Serpent from a group of three academics based in Israel, Norway and the UK,
- Rijndael from a couple of Belgian cryptographers.

Finally in the fall of 2000, NIST announced that the overall AES winner had been chosen to be Rijndael.

DES and all the AES finalists are examples of *iterated* block ciphers. The block ciphers obtain their security by repeated use of a simple *round function*. The round function takes an n-bit block and returns an n-bit block, where n is the block size of the overall cipher. The number of rounds r can either be a variable or fixed. As a general rule increasing the number of rounds will increase the level of security of the block cipher.

Each use of the round function employs a round key

$$k_i \text{ for } 1 \leq i \leq r$$

derived from the main secret key k, using an algorithm called a *key schedule*. To allow decryption, for every round key the function implementing the round must be invertible, and for decryption the round keys are used in the opposite order that they were used for encryption. That the whole round is invertible does not imply that the functions used to implement the round need to be invertible. This may seem strange at first reading but will become clearer when we discuss the DES cipher later. In DES the functions needed to implement the round function are not invertible, but the whole round is invertible. For Rijndael not only is the whole round function invertible but every function used to create the round function is also invertible.

There are a number of general purpose techniques which can be used to break a block cipher, for example: exhaustive search, using pre-computed tables of intermediate values or divide and conquer. Some (badly designed) block ciphers can be susceptible to chosen plaintext attacks, where encrypting a specially chosen plaintext can reveal properties of the underlying secret key. In cryptanalysis one needs a combination of mathematical and puzzle-solving skills, plus luck. There are a few more advanced techniques which can be employed, some of which apply in general to any cipher (and not just a block cipher).

- **Differential Cryptanalysis:** In differential cryptanalysis one looks at ciphertext pairs, where the plaintext has a particular difference. The exclusive-or of such pairs is called a

differential and certain differentials have certain probabilities associated to them, depending on what the key is. By analysing the probabilities of the differentials computed in a chosen plaintext attack one can hope to reveal the underlying structure of the key.

- **Linear Cryptanalysis:** Even though a good block cipher should contain non-linear components the idea behind linear cryptanalysis is to approximate the behaviour of the non-linear components with linear functions. Again the goal is to use a probabilistic analysis to determine information about the key.

Surprisingly these two methods are quite successful against some ciphers. But they do not appear that successful against DES or Rijndael, two of the most important block ciphers in use today.

Since DES and Rijndael are likely to be the most important block ciphers in use for the next few years we shall study them in some detail. This is also important since they both show general design principles in their use of substitutions and permutations. Recall that the historical ciphers made use of such operations, so we see that not much has changed. Now, however, the substitutions and permutations used are far more intricate. On their own they do not produce security, but when used over a number of rounds one can obtain enough security for our applications.

We end this section by discussing which is best, a block cipher or a stream cipher? Alas there is no correct answer to this question. Both have their uses and different properties. Here are just a few general points.

- Block ciphers are more general, and we shall see that one can easily turn a block cipher into a stream cipher.
- Stream ciphers generally have a more mathematical structure. This either makes them easier to break or easier to study to convince oneself that they are secure.
- Stream ciphers are generally not suitable for software, since they usually encrypt one bit at a time. However, stream ciphers are highly efficient in hardware.
- Block ciphers are suitable for both hardware and software, but are not as fast in hardware as stream ciphers.
- Hardware is always faster than software, but this performance improvement comes at the cost of less flexibility.

2. Feistel Ciphers and DES

The DES cipher is a variant of the basic Feistel cipher described in Fig. 2, named after H. Feistel who worked at IBM and performed some of the earliest non-military research on encryption algorithms. The interesting property of a Feistel cipher is that the round function is invertible regardless of the choice of the function in the box marked F. To see this notice that each encryption round is given by

$$\begin{aligned} L_i &= R_{i-1}, \\ R_i &= L_{i-1} \oplus F(K_i, R_{i-1}). \end{aligned}$$

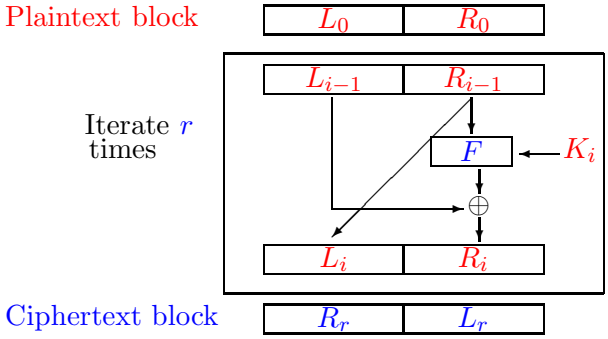
Hence, the decryption can be performed via

$$\begin{aligned} R_{i-1} &= L_i, \\ L_{i-1} &= R_i \oplus F(K_i, L_i). \end{aligned}$$

This means that in a Feistel cipher we have simplified the design somewhat, since

- we can choose any function for the function F, and we will still obtain an encryption function which can be inverted using the secret key,

FIGURE 2. Basic operation of a Feistel cipher



- the same code/circuitry can be used for the encryption and decryption functions. We only need to use the round keys in the reverse order for decryption.

Of course to obtain a secure cipher we still need to take care with

- how the round keys are generated,
- how many rounds to take,
- how the function F is defined.

Work on DES was started in the early 1970s by a team in IBM which included Feistel. It was originally based on an earlier cipher of IBM's called Lucifer, but some of the design was known to have been amended by the National Security Agency, NSA. For many years this led the conspiracy theorists to believe that the NSA had placed a trapdoor into the design of the function F. However, it is now widely accepted that the modifications made by the NSA were done to make the cipher more secure. In particular, the changes made by the NSA made the cipher resistant to differential cryptanalysis, a technique that was not discovered in the open research community until the 1980s.

DES is also known as the Data Encryption Algorithm DEA in documents produced by the American National Standards Institute, ANSI. The International Standards Organisation ISO refers to DES by the name DEA-1. It has been a world-wide standard for well over twenty years and stands as the first publicly available algorithm to have an 'official status'. It therefore marks an important step on the road from cryptography being a purely military area to being a tool for the masses.

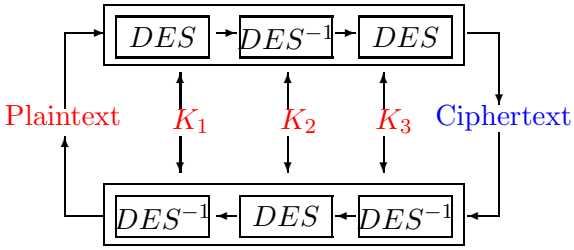
The basic properties of the DES cipher are that it is a variant of the Feistel cipher design with

- the number of rounds r is 16,
- the block length n is 64 bits,
- the key length is 56 bits,
- the round keys $K_1, \ldots, K_{16}$ are each 48 bits.

Note that a key length of 56 bits is insufficient for many modern applications, hence often one uses DES by using three keys and three iterations of the main cipher. Such a version is called Triple DES or 3DES, see Fig. 3. In 3DES the key length is equal to 168. There is another way of using DES three times, but using two keys instead of three giving rise to a key length of 112. In this two-key version of 3DES one uses the 3DES basic structure but with the first and third key being equal. However, two-key 3DES is not as secure as one might inititally think.

2.1. Overview of DES Operation. Basically DES is a Feistel cipher with 16 rounds, as depicted in Fig. 4, except that before and after the main Feistel iteration a permutation is performed. Notice how the two blocks are swapped around before being passed through the final inverse permutation. This permutation appears to produce no change to the security, and people have often wondered why it is there. One answer given by one of the original team members was that this permutation was there to make the original implementation easier to fit on the circuit board.

FIGURE 3. Triple DES

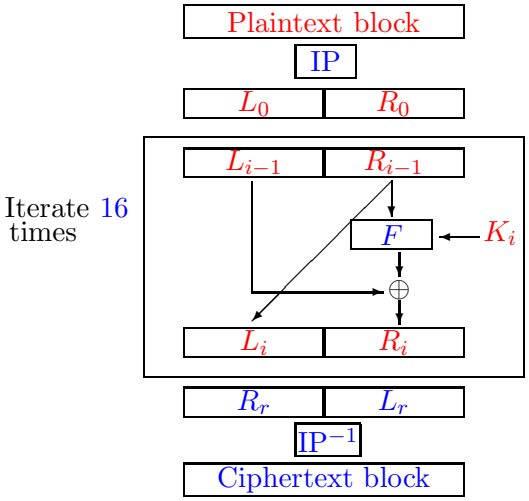


In summary the DES cipher operates on 64 bits of plaintext in the following manner:

- Perform an initial permutation.
- Split the blocks into left and right half.
- Perform 16 rounds of identical operations.
- Join the half blocks back together.
- Perform a final permutation.

The final permutation is the inverse of the initial permutation, this allows the same hardware/software to be used for encryption and decryption. The key schedule provides 16 round keys of 48 bits in length by selecting 48 bits from the 56-bit main key.

FIGURE 4. DES as a Feistel cipher



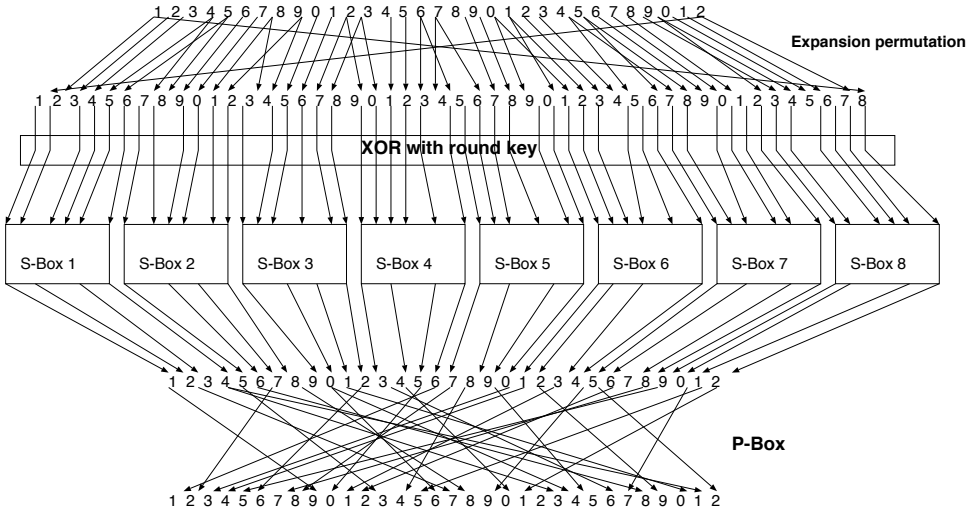
We shall now describe the operation of the function F. In each DES round this consists of the following six stages:

- **Expansion Permutation:** The right half of 32 bits is expanded and permuted to 48 bits. This helps the diffusion of any relationship of input bits to output bits. The expansion permutation (which is different from the initial permutation) has been chosen so that one bit of input affects two substitutions in the output, via the S-Boxes below. This helps spread dependencies and creates an avalanche effect (a small difference between two plaintexts will produce a very large difference in the corresponding ciphertexts).
- **Round Key Addition:** The 48-bit output from the expansion permutation is XORed with the round key, which is also 48 bits in length. Note, this is the only place where the round key is used in the algorithm.
- **Splitting:** The resulting 48-bit value is split into eight lots of six-bit values.

- **S-Box:** Each six-bit value is passed into one of eight different S-Boxes (Substitution Box) to produce a four-bit result. The S-Boxes represent the non-linear component in the DES algorithm and their design is a major contributor to the algorithms security. Each S-Box is a look-up table of four rows and sixteen columns. The six input bits specify which row and column to use. Bits 1 and 6 generate the row number, whilst bits $2, 3, 4$ and 5 specify the column number. The output of each S-Box is the value held in that element in the table.
- **P-Box:** We now have eight lots of four-bit outputs which are then combined into a 32-bit value and permuted to form the output of the function F.

The overall structure of DES is explained in Fig. 5.

FIGURE 5. Structure of the DES function F



We now give details of each of the steps which we have not yet fully defined.

2.1.1. *Initial Permutation, IP:.* The DES initial permutation is defined in the following table. Here the 58 in the first position means that the first bit of the output from the IP is the 58th bit of the input, and so on.

58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

The inverse permutation is given in a similar manner by the following table.

40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25

2.1.2. *Expansion Permutation, E:.* The expansion permutation is given in the following table. Each row corresponds to the bits which are input into the corresponding S-Box at the next stage. Notice how the bits which select a row of one S-Box (the first and last bit on each row) are also used to select the column of another S-Box.

32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

2.1.3. *S-Box:* The details of the eight DES S-Boxes are given in the Fig. 6. Recall that each box consists of a table with four rows and sixteen columns.

2.1.4. *The P-Box Permutation, P:.* The P-Box permutation takes the eight lots of four-bit nibbles, output of the S-Boxes, and produces a 32-bit permutation of these values as given by the following table.

16	7	20	21
29	12	28	17
1	15	23	26
5	18	31	10
2	8	24	14
32	27	3	9
19	13	30	6
22	11	4	25

2.2. DES Key Schedule. The DES key schedule takes the 56-bit key, which is actually input as a bitstring of 64 bits comprising of the key and eight parity bits, for error detection. These parity bits are in bit positions $8, 16, \ldots, 64$ and ensure that each byte of the key contains an odd number of bits.

We first permute the bits of the key according to the following permutation (which takes a 64-bit input and produces a 56-bit output, hence discarding the parity bits).

57	49	41	33	25	17	9
1	58	50	42	34	26	18
10	2	59	51	43	35	27
19	11	3	60	52	44	36
63	55	47	39	31	23	15
7	62	54	46	38	30	22
14	6	61	53	45	37	29
21	13	5	28	20	12	4

FIGURE 6. DES S-Boxes

S-Box 1

14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

S-Box 2

15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9

S-Box 3

10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12

S-Box 4

7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14

S-Box 5

2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3

S-Box 6

12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13

S-Box 7

4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12

S-Box 8

13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

The output of this permutation, called PC-1 in the literature, is divided into a 28-bit left half C_0 and a 28-bit right half D_0. Now for each round we compute

$$C_i = C_{i-1} \lll p_i,$$
$$D_i = D_{i-1} \lll p_i,$$

where $x \lll p_i$ means perform a cyclic shift on x to the left by p_i positions. If the round number i is $1, 2, 9$ or 16 then we shift left by one position, otherwise we shift left by two positions.

Finally the two portions C_i and D_i are joined back together and are subject to another permutation, called PC-2, to produce the final 48-bit round key. The permutation PC-2 is described below.

14	17	11	24	1	5
3	28	15	6	21	10
23	19	12	4	26	8
16	7	27	20	13	2
41	52	31	37	47	55
30	40	51	45	33	48
44	49	39	56	34	53
46	42	50	36	29	32

CIS 331 Note: the section on the Rijndael algorithm (block cipher used in AES) which starts here has been cut. You can still find it in the original book, which can be obtained from Nigel Smart's web page.

4. Modes of Operation

A block cipher like DES or Rijndael can be used in a variety of ways to encrypt a data string. Soon after DES was standardized another US Federal standard appeared giving four *recommended* ways of using DES for data encryption. These modes of operation have since been standardized internationally and can be used with any block cipher. The four modes are

- **ECB Mode:** This is simple to use, but suffers from possible deletion and insertion attacks. A one-bit error in ciphertext gives one whole block error in the decrypted plaintext.
- **CBC Mode:** This is the best mode to use as a block cipher since it helps protect against deletion and insertion attacks. In this mode a one-bit error in the ciphertext gives not only a one-block error in the corresponding plaintext block but also a one-bit error in the next decrypted plaintext block.

we could simply delete blocks from the message and no one would know. Thirdly we could replay known blocks from other messages. By extracting ciphertext corresponding to a known piece of plaintext we can then amend other transactions to contain this known block of text.

To see all these problems suppose our block cipher is rather simple and encrypts each English word as a block. Suppose we obtained the encryption of the sentences

Pay Alice one hundred pounds,
Don't pay Bob two hundred pounds,

which encrypted were

the horse has four legs,
stop the pony hasn't four legs.

We can now make the recipient pay Alice two hundred pounds by sending her the message

the horse hasn't four legs,

in other words we have replaced a block from one message by a block from another message. Or we could stop the recipient paying Alice one hundred pounds by inserting the encryption stop of don't onto the front of the original message to Alice. Or we can make the recipient pay Bob two hundred pounds by deleting the first block of the message sent to him.

These threats can be countered by adding checksums over a number of plaintext blocks, or by using a mode of operation which adds some 'context' to each ciphertext block.

4.2. CBC Mode. One way of countering the problems with ECB Mode is to chain the cipher, and in this way add context to each ciphertext block. The easiest way of doing this is to use Cipher Block Chaining Mode, or CBC Mode.

Again, the plaintext must first be divided into a series of blocks

$$m_1, \ldots, m_q,$$

and as before the final block may need padding to make the plaintext length a multiple of the block length. Encryption is then performed via the equations

$$\begin{aligned} c_1 &= e_k(m_1 \oplus IV), \\ c_i &= e_k(m_i \oplus c_{i-1}) \text{ for } i > 1, \end{aligned}$$

see also Fig. 9.

Notice that we require an additional initial value IV to be passed to the encryption function, which can be used to make sure that two encryptions of the same plaintext produce different ciphertexts. In some situations one therefore uses a random IV with every message, usually when the same key will be used to encrypt a number of messages. In other situations, mainly when the key to the block cipher is only going to be used once, one chooses a fixed IV, for example the all zero string. In the case where a random IV is used, it is not necessary for the IV to be kept secret and it is usually transmitted in the clear from the encryptor to the decryptor as part of the message. The distinction between the reasons for using a fixed or random value for IV is expanded upon further in Chapter 21.

Decryption also requires the IV and is performed via the equations,

$$\begin{aligned} m_1 &= d_k(c_1) \oplus IV, \\ m_i &= d_k(c_i) \oplus c_{i-1} \text{ for } i > 1, \end{aligned}$$

see Fig. 10.

With ECB Mode a single bit error in transmission of the ciphertext will result in a whole block being decrypted wrongly, whilst in CBC Mode we see that not only will we decrypt a block incorrectly but the error will also affect a single bit of the next block.

FIGURE 9. CBC encipherment

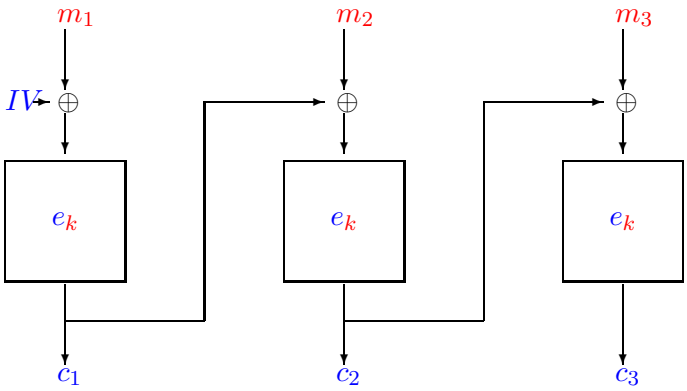
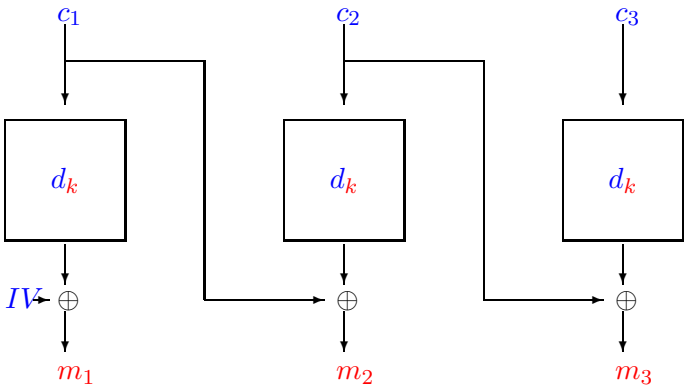


FIGURE 10. CBC decipherment



4.3. OFB Mode. Output Feedback Mode, or OFB Mode enables a block cipher to be used as a stream cipher. We need to choose a variable j $(1 \leq j \leq n)$ which will denote the number of bits output by the keystream generator on each iteration. We use the block cipher to create the keystream, j bits at a time. It is however usually recommended to take $j = n$ as that makes the expected cycle length of the keystream generator larger.

Again we divide plaintext into a series of blocks, but this time each block is j-bits, rather than n-bits long:

$$m_1, \ldots, m_q.$$

Encryption is performed as follows, see Fig. 11 for a graphical representation. First we set $X_1 = IV$, then for $i = 1, 2, \ldots, q$, we perform the following steps,

$$\begin{aligned} Y_i &= e_k(X_i), \\ E_i &= j \text{ leftmost bits of } Y_i, \\ c_i &= m_i \oplus E_i, \\ X_{i+1} &= Y_i. \end{aligned}$$

Decipherment in OFB Mode is performed in a similar manner as described in Fig. 12.

4.4. CFB Mode. The next mode we consider is called Cipher FeedBack Mode, or CFB Mode. This is very similar to OFB Mode in that we use the block cipher to produce a stream cipher. Recall

FIGURE 11. OFB encipherment

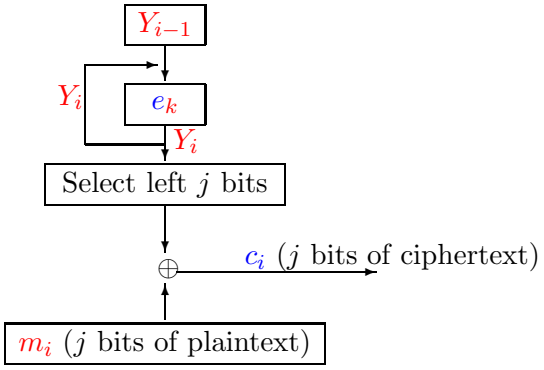
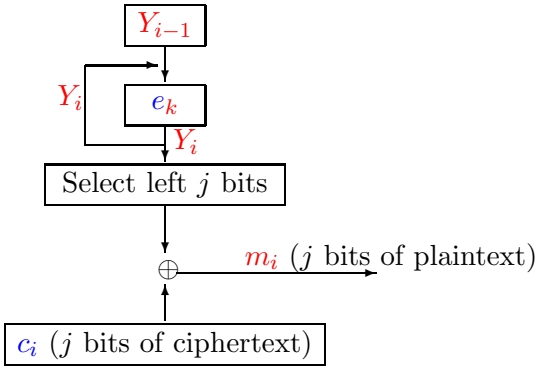


FIGURE 12. OFB decipherment



that in OFB Mode the keystream was generated by encrypting the IV and then iteratively encrypting the output from the previous encryption. In CFB Mode the keystream output is produced by the encryption of the ciphertext, as in Fig. 13, by the following steps,

$$\begin{aligned} Y_0 &= IV, \\ Z_i &= e_k(Y_{i-1}), \\ E_i &= j \text{ leftmost bits of } Z_i, \\ Y_i &= m_i \oplus E_i. \end{aligned}$$

We do not present the decryption steps, but leave these as an exercise for the reader.

4.5. CTR Mode. The next mode we consider is called Counter Mode, or CTR Mode. This combines many of the advantages of ECB Mode, but with none of the disadvantages. We first select a public IV, or counter, which is chosen differently for each message encrypted under the fixed key k. Then encryption proceeds for the ith block, by encrypting the value of $IV + i$ and then xor'ing this with the message block. In other words we have

$$c_i = m_i \oplus e_k(IV + i).$$

This is explained pictorially in Figure 14

CTR Mode has a number of interesting properties. Firstly since each block can be encrypted independently, much like in ECB Mode, we can process each block at the same time. Compare this to CBC Mode, OFB Mode or CFB Mode where we cannot start encrypting the second block until the first block has been encrypted. This means that encryption, and decryption, can be performed in parallel. However, unlike ECB Mode two equal blocks will not encrypt to the same ciphertext

FIGURE 13. CFB encipherment

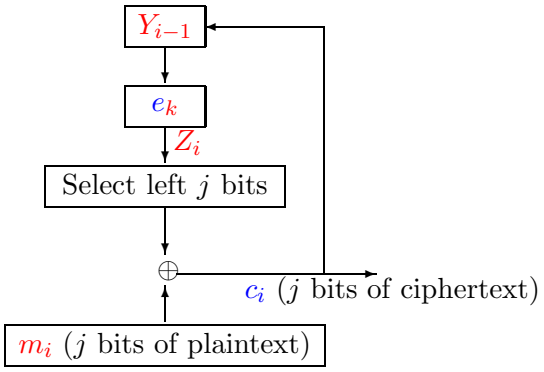
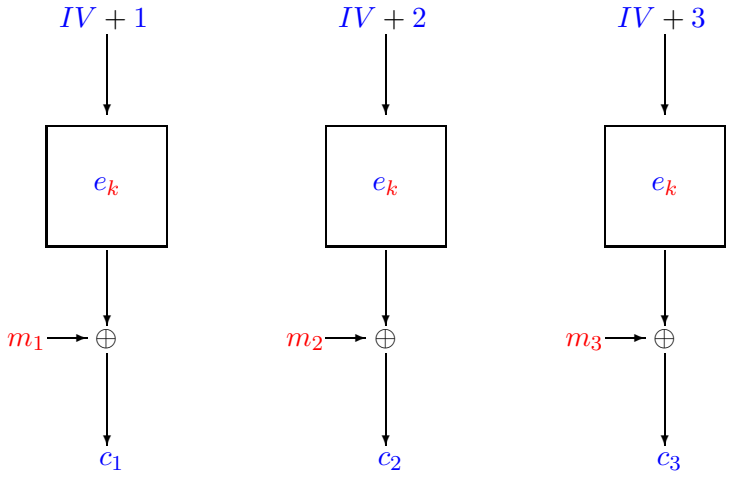


FIGURE 14. CTR encipherment



value. This is because each plaintext block is encrypted using a different input to the encryption function, in some sense we are using the block cipher encryption of the different inputs to produce a stream cipher. Also unlike ECB Mode each ciphertext block corresponds to a precise position within the ciphertext, as its position information is needed to be able to decrypt it successfully.

Chapter Summary

- The most popular block cipher is DES, which is itself based on a general design called a Feistel cipher.
- A comparatively recent block cipher is the AES cipher, called Rijndael.
- Both DES and Rijndael obtain their security by repeated application of simple rounds consisting of substitution, permutation and key addition.
- To use a block cipher one needs to also specify a mode of operation. The simplest mode is ECB mode, which has a number of problems associated with it. Hence, it is common to use a more advanced mode such as CBC or CTR mode.

- Some block cipher modes, such as CFB, OFB and CTR modes, allow the block cipher to be used in a stream cipher.

Further Reading

The Rijndael algorithm, the AES process and a detailed discussion of attacks on block ciphers and Rijndael in particular can be found in the book by Daemen and Rijmen. Stinson's book is the best book to explain differential cryptanalysis for students.

J. Daemen and V. Rijmen. *The Design of Rijndael: AES – The Advanced Encryption Standard.* Springer-Verlag, 2002.

D. Stinson. *Cryptography Theory and Practice.* CRC Press, 1995.