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

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

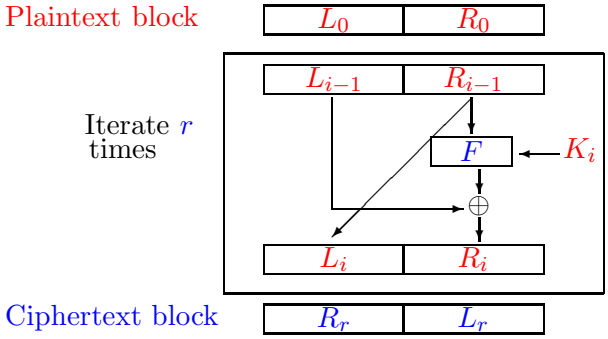
Hence, the decryption can be performed via

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

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.

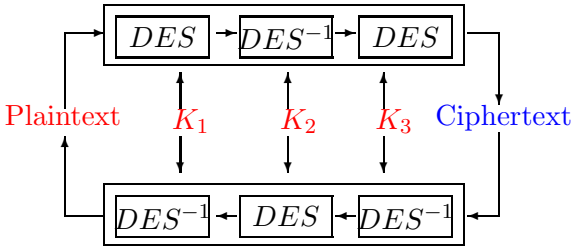
The basic properties of the DES cipher are that it is a variant of the Feistel cipher design with

- 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

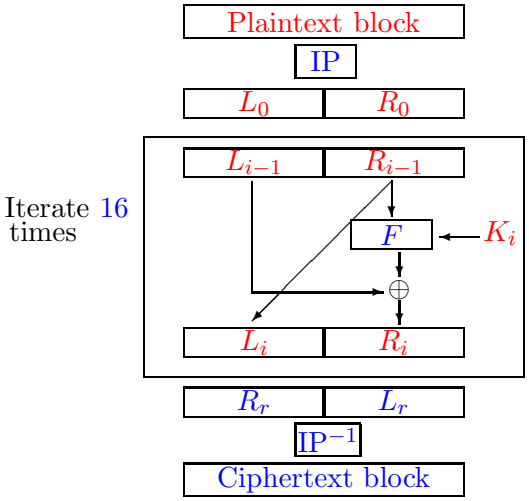


In summary the DES cipher operates on 64 bits of plaintext in the following manner:

- 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



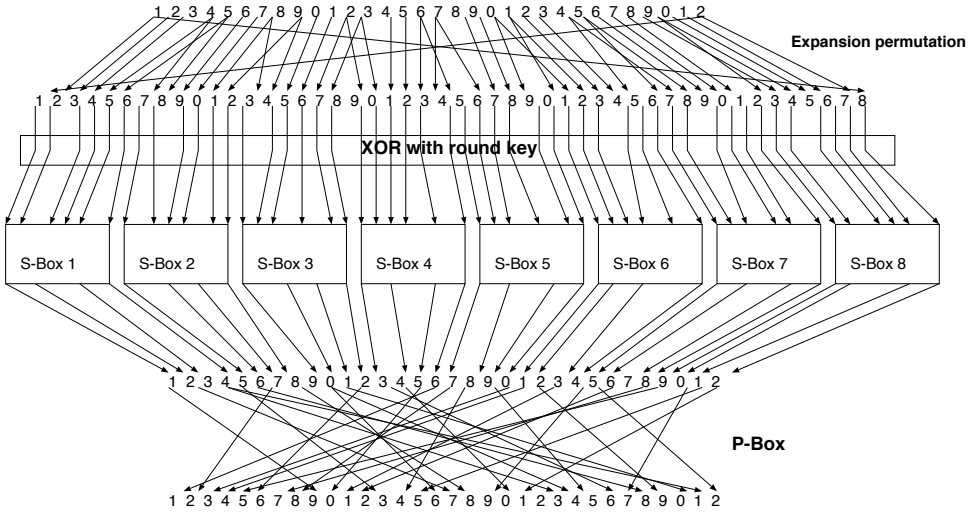
We shall now describe the operation of the function F. In each DES round this consists of the following six stages:

- **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

3. Rijndael

The AES winner was decided in fall 2000 to be the Rijndael algorithm designed by Daemen and Rijmen. Rijndael is a block cipher which does not rely on the basic design of the Feistel cipher. However, Rijndael does have a number of similarities with DES. It uses a repeated number of rounds to obtain security and each round consists of substitutions and permutations, plus a key addition phase. Rijndael in addition has a strong mathematical structure, as most of its operations are based on arithmetic in the field $\mathbb{F}_{2^8}$. However, unlike DES the encryption and decryption operations are distinct.

Recall that elements of $\mathbb{F}_{2^8}$ are stored as bit vectors (or bytes) representing binary polynomials. For example the byte given by $0x83$ in hexadecimal, gives the bit pattern

$$1, 0, 0, 0, 0, 0, 1, 1$$

since

$$0x83 = 8 \cdot 16 + 3 = 131$$

in decimal. One can obtain the bit pattern directly by noticing that 8 in binary is $1, 0, 0, 0$ and 3 in 4-bit binary is $0, 0, 1, 1$ and one simply concatenates these two bit strings together. The bit pattern itself then corresponds to the binary polynomial

$$x^7 + x + 1.$$

So we say that the hexadecimal number $0x83$ represents the binary polynomial

$$x^7 + x + 1.$$

Arithmetic in $\mathbb{F}_{2^8}$ is performed using polynomial arithmetic modulo the irreducible polynomial

$$m(x) = x^8 + x^4 + x^3 + x + 1.$$

Rijndael identifies 32-bit words with polynomials in $\mathbb{F}_{2^8}[X]$ of degree less than four. This is done in a big-endian format, in that the smallest index corresponds to the least important coefficient. Hence, the word

$$a_0 \| a_1 \| a_2 \| a_3$$

will correspond to the polynomial

$$a_3 X^3 + a_2 X^2 + a_1 X + a_0.$$

Arithmetic is performed on polynomials in $\mathbb{F}_{2^8}[X]$ modulo the reducible polynomial

$$M(X) = X^4 + 1.$$

Hence, arithmetic is done on these polynomials in a ring rather than a field, since $M(X)$ is reducible.

Rijndael is a parametrized algorithm in that it can operate on block sizes of 128, 192 or 256 bits, it can also accept keys of size 128, 192 or 256 bits. For each combination of block and key size a different number of rounds is specified. To make our discussion simpler we shall consider