

UNIT-2

The Feistel Cipher:

Feistel proposed [FEIS73] that we can approximate the ideal block cipher by utilizing the concept of a product cipher, which is the execution of two or more simple ciphers in sequence in such a way that the final result or product is cryptographically stronger than any of the component ciphers. The essence of the approach is to develop a block cipher with a key length of k bits and a block length of n bits, allowing a total of 2^k possible transformations, rather than the $2^n!$ transformations available with the ideal block cipher.

In particular, Feistel proposed the use of a cipher that alternates substitutions and permutations, where these terms are defined as follows:

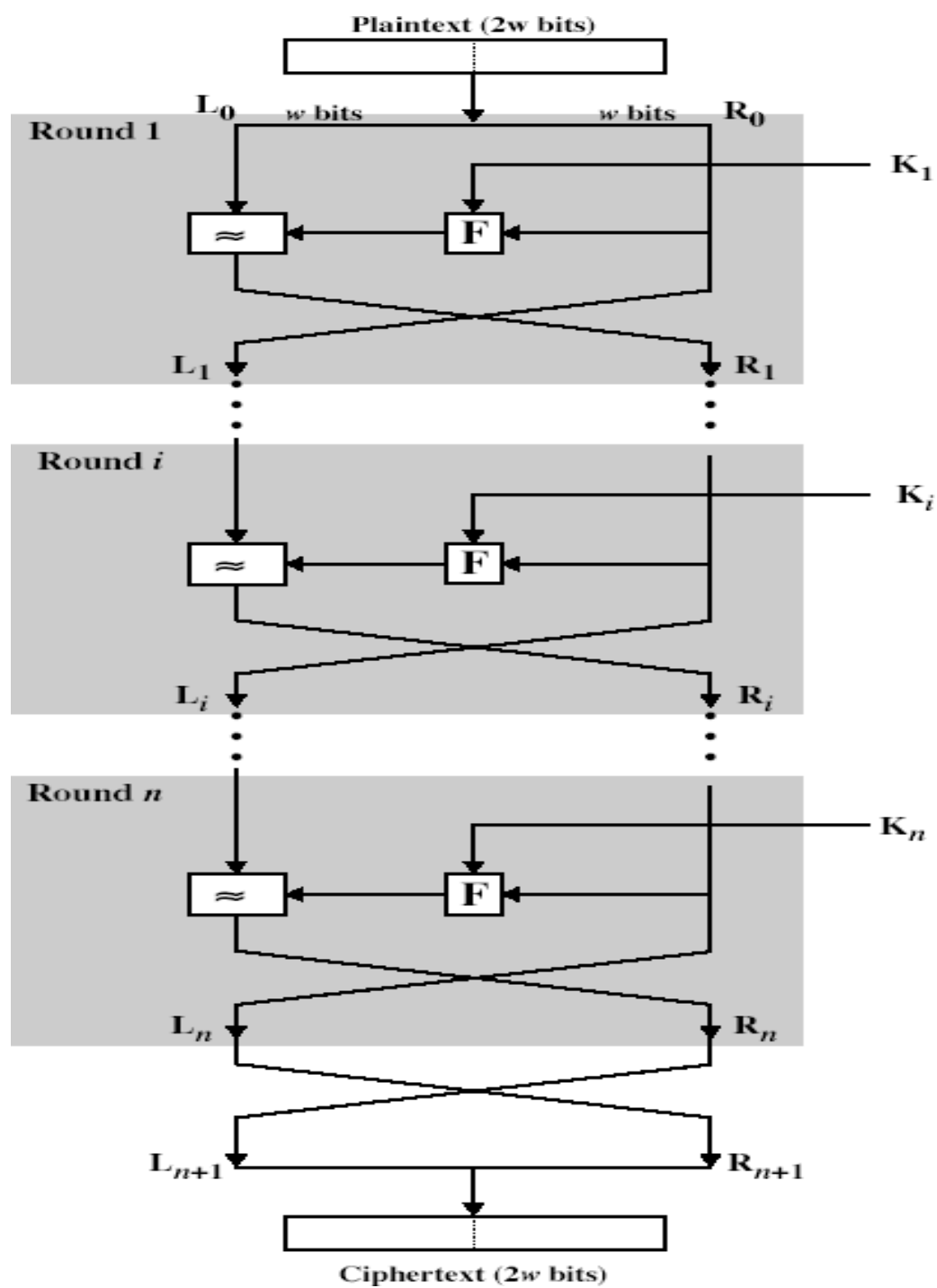
- **Substitution:** Each plaintext element or group of elements is uniquely replaced by a corresponding ciphertext element or group of elements.
- **Permutation:** A sequence of plaintext elements is replaced by a permutation of that sequence. That is, no elements are added or deleted or replaced in the sequence, rather the order in which the elements appear in the sequence is changed.

Feistel's is a practical application of a proposal by Claude Shannon to develop a product cipher that alternates *confusion* and *diffusion* functions

FEISTEL CIPHER STRUCTURE The left-hand side of Figure 3.3 depicts the structure proposed by Feistel. The inputs to the encryption algorithm are a plaintext block of length $2w$ bits and a key. The plaintext block is divided into two halves, L_0 and R_0 . The two halves of the data pass through n rounds of processing and then combine to produce the ciphertext block. Each round i has as inputs L_{i-1} and R_{i-1} derived from the previous round, as well as a subkey K_i derived from the overall K . In general, the subkeys K_i are different from K and from each other.

All rounds have the same structure. A substitution is performed on the left half of the data. This is done by applying a round function F to the right half of the data and then taking the exclusive-OR of the output of that function and the left half of the data. The round function has the same general structure for each round but is parameterized by the round subkey K_i .

Feistel Cipher structures



Permutation is performed that consists of the interchange of the two halves of the data. This structure is a particular form of the substitution-permutation network (SPN) proposed by Shannon.

The exact realization of a Feistel network depends on the choice of the following parameters and design features:

Block size: Larger block sizes mean greater security (all other things being equal) but reduced encryption/decryption speed for a given algorithm. The greater security is achieved by greater diffusion. Traditionally, a block size of 64 bits has been considered a reasonable tradeoff and was nearly universal in block cipher design. However, the new AES uses a 128-bit block size.

Key size: Larger key size means greater security but may decrease encryption/ decryption speed. The greater security is achieved by greater resistance to brute-force attacks and greater confusion. Key sizes of 64 bits or less are now widely considered to be inadequate, and 128 bits has become a common size.

Number of rounds: The essence of the Feistel cipher is that a single round offers inadequate security but that multiple rounds offer increasing security. A typical size is 16 rounds.

Subkey generation algorithm: Greater complexity in this algorithm should lead to greater difficulty of cryptanalysis.

There are two other considerations in the design of a Feistel cipher:

Fast software encryption/decryption: In many cases, encryption is embedded in applications or utility functions in such a way as to preclude a hardware implementation. Accordingly, the speed of execution of the algorithm becomes a concern.

Round function F: Again, greater complexity generally means greater resistance to cryptanalysis.

Ease of analysis: Although we would like to make our algorithm as difficult as possible to cryptanalyze, there is great benefit in making the algorithm easy to analyze. That is, if the algorithm can be concisely and clearly explained, it is easier to analyze that algorithm for cryptanalytic vulnerabilities and therefore develop a higher level of assurance as to its strength.

DES,

for example, does not have an easily analyzed functionality

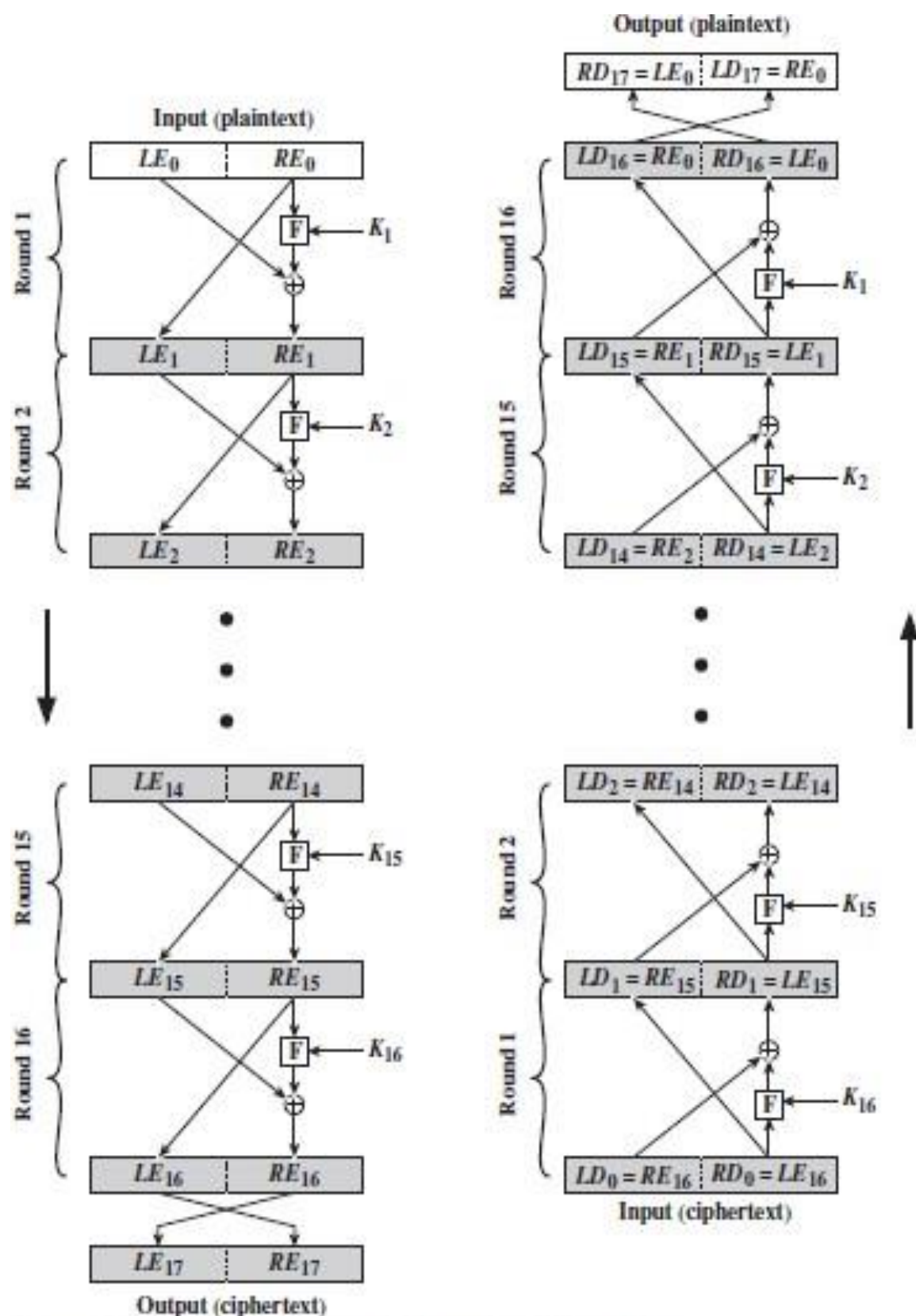
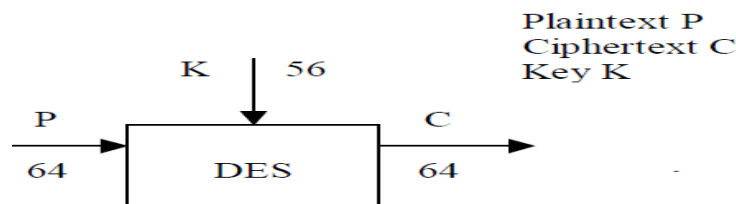


Figure 3.3 Feistel Encryption and Decryption (16 rounds)

<p>encryption process</p> $LE_{16} = RE_{15}$ $RE_{16} = LE_{15} \oplus F(RE_{15}, K_{16})$ <p>in general</p> $LE_i = RE_{i-1}$ $RE_i = LE_{i-1} \oplus F(RE_{i-1}, K_i)$	<p>On the decryption side,</p> $LD_1 = RD_0 = LE_{16} = RE_{15}$ $RD_1 = LD_0 \oplus F(RD_0, K_{16})$ $= RE_{16} \oplus F(RE_{15}, K_{16})$ $= [LE_{15} \oplus F(RE_{15}, K_{16})] \oplus F(RE_{15}, K_{16})$
---	---

Data Encryption Standard (DES):

- DES is a Symmetric-key algorithm for the encryption of electronic data.
- DES originated at IBM in 1977 & was adopted by the U.S Department of Defence. Now it is under the NIST (National Institute of Standard & Technology)
- Data Encryption Standard (DES) is a widely-used method of data encryption using a private (secret) key
- DES applies a 56-bit key to each 64-bit block of data. The process can run in several modes and involves 16 rounds or operations.



Inner workings of DES:

DES (and most of the other major symmetric ciphers) is based on a cipher known as the Feistel block cipher. This was a block cipher developed by the IBM cryptography researcher Horst Feistel in the early 70's. It consists of a number of rounds where each round contains bit-shuffling, non-linear substitutions (S-boxes) and exclusive OR operations. Most symmetric encryption schemes today are based on this structure (known as a feistel network).

Overall structure

DES (and most of the other major symmetric ciphers) is based on a cipher known as the Feistel block cipher.

Looking at the left-hand side of the figure, we can see that the processing of the plaintext proceeds in three phases.

- First, the 64-bit plaintext passes through an initial permutation (IP) that rearranges the bits to produce the *permuted input*.
- This is followed by a phase consisting of sixteen rounds of the same function, which involves both permutation and substitution functions. The output of the last (sixteenth) round consists of 64 bits that are a function of the input plaintext and the key. The left and right halves of the output are swapped to produce the **preoutput**.
- Finally, the preoutput is passed through a permutation that is the inverse of the initial permutation function, to produce the 64-bit cipher text. With the exception of the initial and final permutations, DES has the exact structure of a Feistel cipher,

Table 3.2 Permutation Tables for DES

(a) Initial Permutation (IP)

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

(b) Inverse Initial Permutation (IP^{-1})

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

(c) Expansion Permutation (E)

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

(d) Permutation Function (P)

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

To see that these two permutation functions are indeed the inverse of each other, consider the following 64-bit input M:

M_1 M_2 M_3 M_4 M_5 M_6 M_7 M_8
 M_9 M_{10} M_{11} M_{12} M_{13} M_{14} M_{15} M_{16}
 M_{17} M_{18} M_{19} M_{20} M_{21} M_{22} M_{23} M_{24}
 M_{25} M_{26} M_{27} M_{28} M_{29} M_{30} M_{31} M_{32}
 M_{33} M_{34} M_{35} M_{36} M_{37} M_{38} M_{39} M_{40}
 M_{41} M_{42} M_{43} M_{44} M_{45} M_{46} M_{47} M_{48}
 M_{49} M_{50} M_{51} M_{52} M_{53} M_{54} M_{55} M_{56}
 M_{57} M_{58} M_{59} M_{60} M_{61} M_{62} M_{63} M_{64}

Where M_i is a binary digit. Then the permutation $X = (IP(M))$ is as follows:

The round key K_i is 48 bits. The R input is 32 bits. This R input is first expanded to 48 bits by using a table that defines a permutation plus an expansion that involves duplication of 16 of the R bits (Table 3.2c). The resulting 48 bits are XORed with K_i . This 48-bit result passes through a substitution function that produces a 32-bit output, which is permuted as defined by Table 3.2d. The role of the S-boxes in the function F is illustrated in Figure 3.7. The substitution consists of a set of eight S-boxes, each of which accepts 6 bits as input and produces 4 bits as output. These transformations are defined in Table 3.3, which is interpreted as follows: The first and last bits of the input to box S_i form a 2-bit binary number to select one of four substitutions defined by the four rows in the table for S_i . The middle four bits select one of the sixteen columns. The decimal value in the cell selected by the row and column is then converted to its 4-bit representation to produce the output.

For example, in S_1 , for input 011001, the row is 01 (row 1) and the column is 1100 (column 12). The value in row 1, column 12 is 9, so the output is 1001. Each row of an S-box defines a general reversible substitution. Figure 3.2 may be useful in understanding the mapping. The figure shows the substitution for row 0 of box S_1 . The operation of the S-boxes is worth further comment. Ignore for the moment the contribution of the key (K_i). If you examine the expansion table, you see that the 32 bits of input are split into groups of 4 bits and then become groups of 6 bits by taking the outer bits from the two adjacent groups. For example,

... efgh ijkl mnop ...

This becomes

... defghi hijklm lnopq ...

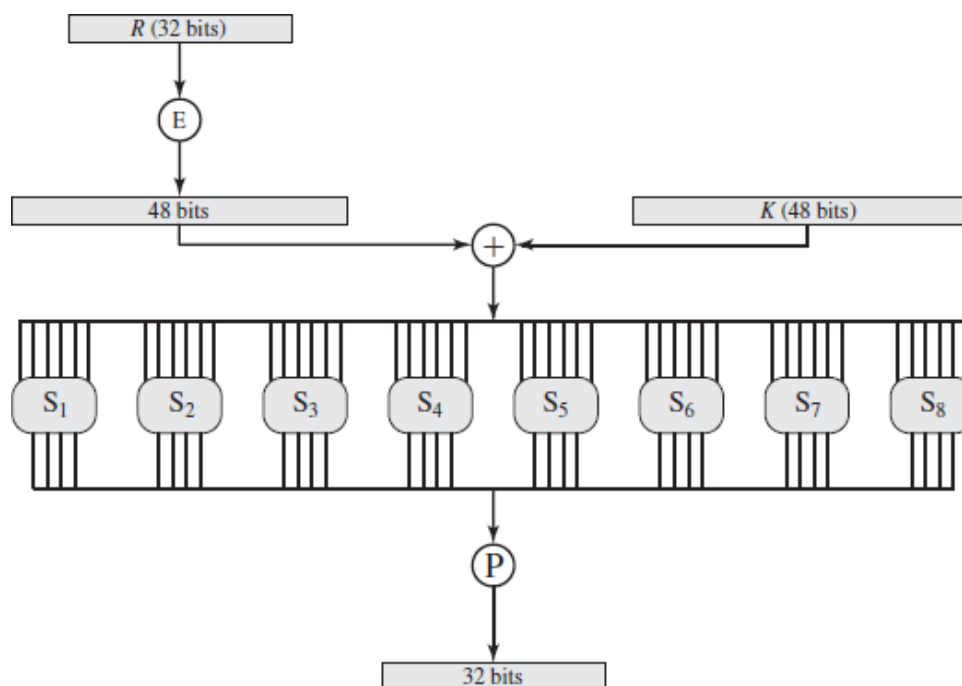


Figure 3.7 Calculation of $F(R, K)$

The outer two bits of each group select one of four possible substitutions (one row of an S-box). Then a 4-bit output value is substituted for the particular 4-bit input (the middle four input bits). The 32-bit output from the eight S-boxes is then permuted, so that on the next round, the output from each S-box immediately affects as many others as possible.

Substitution Boxes S: Have eight S-boxes which map 6 to 4 bits. Each S-box is actually 4 little 4 bit boxes. Outer bits 1 & 6 (**row** bits) select one rows. inner bits 2-5 (**col** bits) are substituted. Result is 8 lots of 4 bits, or 32 bits. Row selection depends on both data & key

KEY GENERATION Returning to above all figures, we see that a 64-bit key is used as input to the algorithm. The bits of the key are numbered from 1 through 64; every eighth bit is ignored, as indicated by the lack of shading in Table 3.4a. The key is first subjected to a permutation governed by a table labeled Permuted Choice One (Table 3.4b)

The resulting 56-bit key is then treated as two 28-bit quantities, labelled C0 and D0. At each round, C_{i-1} and D_{i-1} are separately subjected to a circular left shift or (rotation) of 1 or 2 bits, as governed by Table 3.4d. These shifted values serve as input to the next round. They also serve as input to the part labeled Permuted Choice Two (Table 3.4c), which produces a 48-bit output that serves as input to the Function F(R_{i-1}, K_i).

DES DECRYPTION:

Whatever process we following in the encryption that process is used for decryption also but the order of key is changed on input message (cipher text).

Reverse order of keys are K₁₆, K₁₅,, K₁.

Strengths of DES:

The DES is a symmetric key block cipher which takes 64bits cipher text and 56 bit key as an input and produce 64 bits cipher text as output.

The DES function is made up of P & S boxes

P-boxes transpose bits

S-boxes Substitution bits to generating the cipher text.

The use of 56bits keys: 56 bit key is used in encryption, there are 2^{56} possible keys.

$2^{56} = 7.2 \times 10^{16}$ keys, by this a brute force attack on such number of keys is impractical. A machine performing one DES encryption per microsecond would take more than a thousand years to break the cipher.

The nature of algorithm: Cryptanalyst can perform cryptanalysis by exploiting the characteristic of DES algorithm but no one has succeeded in finding out the weakness. This is possible because, in DES, they using 8-substitution tables or S-boxes in each iteration & one P-box transition for the every individual iteration.

Avalanche Effect:

- key desirable property of an encryption algorithm
- where a change of **one** input or key bit results in changing approx **half** output bits
- making attempts to “home-in” by guessing keys impossible
- DES exhibits strong avalanche

Timing Attacks

- attacks actual implementation of cipher
- use knowledge of consequences of implementation to derive knowledge of some/all subkey bits
- specifically use fact that calculations can take varying times depending on the value of the inputs to it
- particularly problematic on smartcards

Analytic Attacks

- now have several analytic attacks on DES
- these utilize some deep structure of the cipher
 - by gathering information about encryptions
 - can eventually recover some/all of the sub-key bits
 - if necessary then exhaustively search for the rest



- generally these are statistical attacks
- include
 - differential cryptanalysis
 - linear cryptanalysis
 - related key attacks

