Lecture 3: Block Ciphers and the Data Encryption Standard

Lecture Notes on "Introduction to Computer Security"

by Avi Kak (kak@purdue.edu)

January 18, 2007

Goals:

- To introduce the notion a block cipher in the modern context.
- To talk about the infeasibility of **ideal block ciphers**
- To introduce the notion of the **Feistel Cipher Structure**
- To go over **DES**, the Data Encryption Standard

Ideal Block Cipher

- In a modern block cipher (but still using a classical encryption method), we replace a block of N bits from the plaintext with a block of N bits from the ciphertext. This general idea is illustrated in the figure on the next page.

- One typically wants to use 64-bit blocks.

- In an ideal block cipher, the relationship between the input blocks and the output block is completely random. But it must be invertible for decryption to work. Therefore, it has to be one-to-one, meaning that each input block is mapped to a unique output block.

- The encryption key for the ideal block cipher is the codebook itself, meaning the table that shows the relationship between the input blocks and the output blocks.

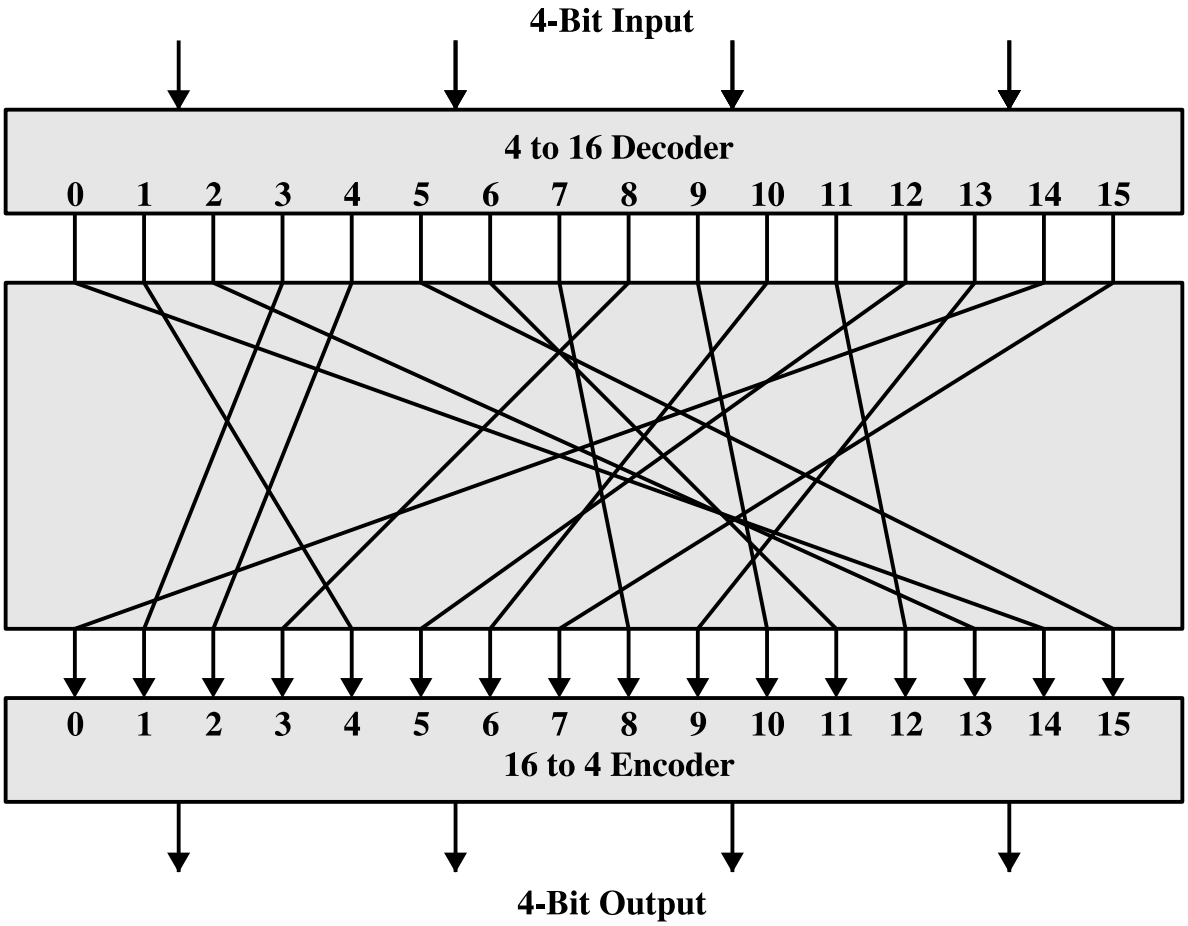

Figure 3.1 General *n*-bit-*n*-bit Block Substitution (shown with *n* = 4)

This figure is from Chapter 3 of Stallings: "Cryptography and Network Security", Fourth Edition

What's the Size of the Encryption Key for the Ideal Block Cipher?

- With a 64-bit block, we can think of each possible input block as one of 2^{64} integers and for each such integer we can specify an output 64-bit block. We can construct the codebook by displaying just the output blocks in the order of the integers corresponding to the input blocks. Such a code book will be of size $64 \times 2^{64} \approx 10^{21}$.

- That implies that the encryption key for the ideal block cipher using 64-bit blocks will be of size 10^{21}.

- The size of the encryption key would make the ideal block cipher an impractical idea.

The Feistel Structure for Block Ciphers

- Consists of multiple rounds of processing of the plaintext, with each round consisting of a substitution step followed by a permutation step.

- The input block to each round is divided into two halves that we can denote **L** and **R** for the left half and the right half.

- In each round, the right half of the block, **R**, goes through unchanged. But the left half, **L**, goes through a **substitution** operation that depends on **R** and the encryption key.

- The permutation step at the end of each round consists of swapping the modified **L** and **R**. *Therefore, the* **L** *for the next round would be* **R** *of the current round. And* **R** *for the next round be the output* **L** *of the current round.*

- These steps are illustrated pictorially on the next page.

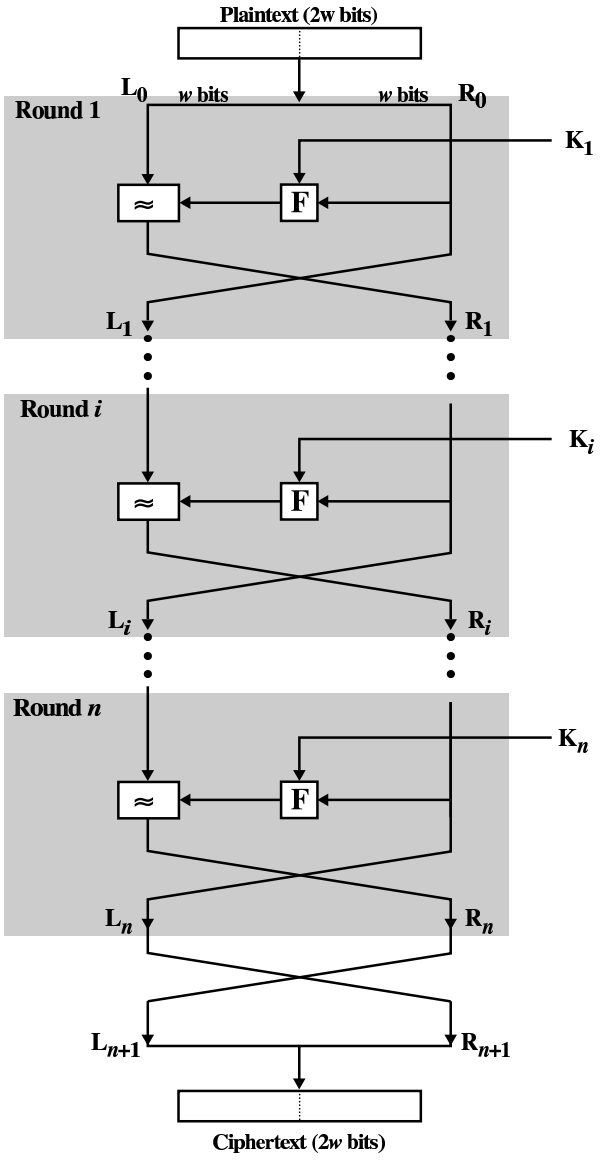

Figure 3.2 Classical Feistel Network

This figure is from Chapter 3 of Stallings: "Cryptography and Network Security", Fourth Edition

Mathematical Description of Each Round in the Feistel Structure

- Let LE_i and RE_i denote the output half-blocks at the end of the i^{th} round of processing. The letter 'E' denotes encryption.

- We obviously have

$$\begin{array}{lcl} LE_i & = & RE_{i-1} \\ RE_i & = & LE_{i-1} \otimes F(RE_{i-1}, K_i) \end{array}$$

where $\otimes$ denotes the bitwise EXCLUSIVE OR operation, $\mathbf{F}$ the substitution function, and K_i the sub-key (derived from the main encryption key) used in the i^{th} round.

- Assuming 16 rounds of processing (which is typical), the output of the last round of processing is given by

$$\begin{array}{lcl} LE_{16} & = & RE_{15} \\ RE_{16} & = & LE_{15} \otimes F(RE_{15}, K_{16}) \end{array}$$

Decryption for Feistel Cipher

- As shown on page 10, the decryption algorithm is exactly the same as the encryption algorithm exception that the data flows through the various rounds in the opposite direction.

- The output of each round during decryption is the input to the corresponding round during encryption. To prove this, let LD_i and RD_i denote the left half and the right half of the output of the i^{th} round.

Proof is simple. Skipped.

- That means that the output of the first decryption round consists of LD_1 and RD_1. So we can denote the input to the first decryption round by LD_0 and RD_0. The relationship between the two halves that are input to the first decryption round and what is output by the encryption algorithm is

$$\begin{aligned} LD_0 &= RE_{16} \\ RD_0 &= LE_{16} \end{aligned}$$

- We can write the following equations for the output of the first decryption round

$$\begin{aligned} LD_1 &= RD_0 \\ &= LE_{16} \\ &= RE_{15} \end{aligned}$$

$$\begin{aligned} RD_1 &= LD_0 \otimes F(RD_0, K_{16}) \\ &= RE_{16} \otimes F(LE_{16}, K_{16}) \\ &= [LE_{15} \otimes F(RE_{15}, K_{16})] \otimes F(RE_{15}, K_{16}) \\ &= LE_{15} \end{aligned}$$

  This shows that the output of the first round of decryption is the same as the input to the last stage of the encryption round since we have $LD_1 = RE_{15}$ and $RD_1 = LE_{15}$

- The following equalities are used in the above derivation. Assume that **A**, **B**, and **C** are bit arrays.

$$\begin{aligned} [A \otimes B] \otimes C &= A \otimes [B \otimes C\ ] \\ A \otimes A &= 0 \\ A \otimes 0 &= A \end{aligned}$$

- The above result is independent of the nature of the substitution function **F**.

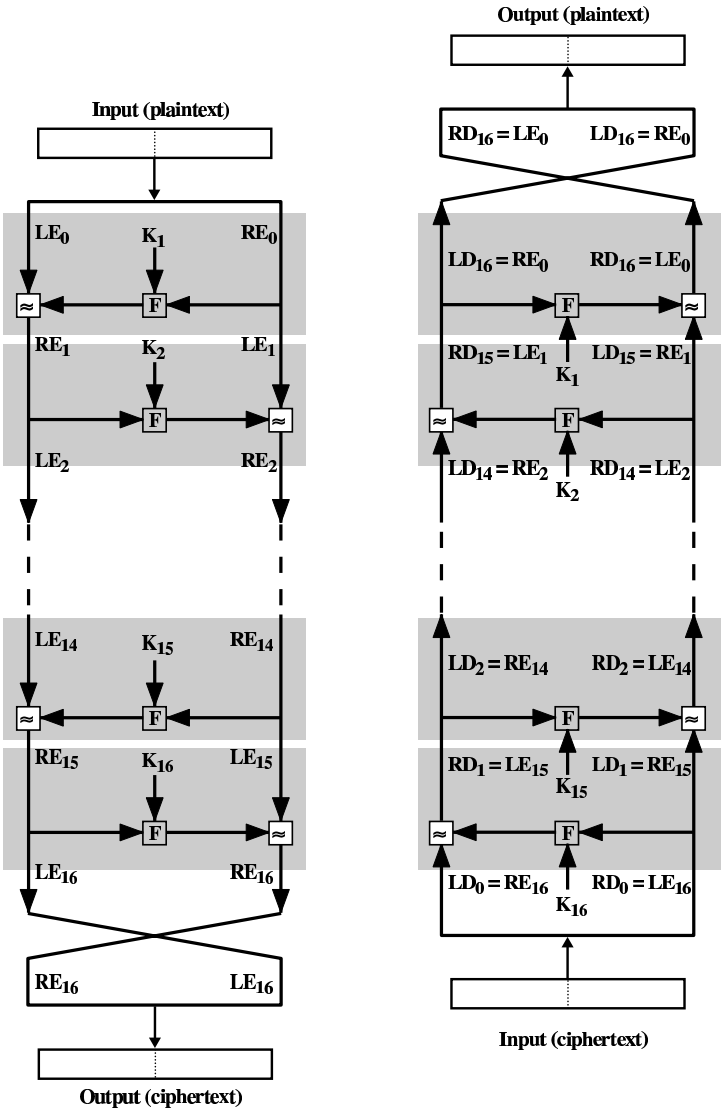

Figure 3.3 Feistel Encryption and Decryption

Encryption is on the left and decryption on the right. **Note that the permutation at the end of each round is now implicit because the direction of the substitution in the alternate rounds is now reversed.**

This figure is from Chapter 3 of Stallings: "Cryptography and Network Security", Fourth Edition

DES: The Data Encryption Standard

- Adopted by NIST in 1977.
- Based on a cipher (Lucifer) developed earlier by IBM for Lloyd's of London for cash transfer.
- DES uses the Feistel cipher structure.
- DES uses a 56-bit encryption key. (The key size was apparently dictated by the memory and processing constraints imposed by a single-chip implementation of the algorithm for DES.)
- DES encryption was broken in 1999 by Electronics Frontiers Organization. This resulted in NIST issuing a new directive that year that required organizations to use **Triple DES**, that is three consecutive applications of **DES**. (That DES was found to be not as strong as originally believed also prompted NIST to initiate the development of new standards for data encryption. The result is **AES** that we will discuss later.)
- **Triple DES** continues to enjoy wide usage in commercial applications. To understand Triple DES, you must first understand the basic **DES** encryption.

DEA: The Data Encryption Algorithm (The Algorithm for DES)

- **DEA** is shown pictorially on the next page.

- The different rounds shown (16 of them) constitute the Feistel cipher structure.

- The data undergoes a permutation step at the beginning (before being fed into the Feistel structure).

- After 16-rounds of processing, an opposite permutation is applied.

- Note how the subkeys for each round are generated from the main key by a sequence of permutations. (Only 56 bits of the 64 bits shown for the main key are relevant. The other 8 bits can be used for parity check, etc.) We will refer to the "Permuted Choice 1" and "Permuted Choice 2" for sub-key generation by PC1 and PC2.

- Note also that each round key is only 48 bits in length.

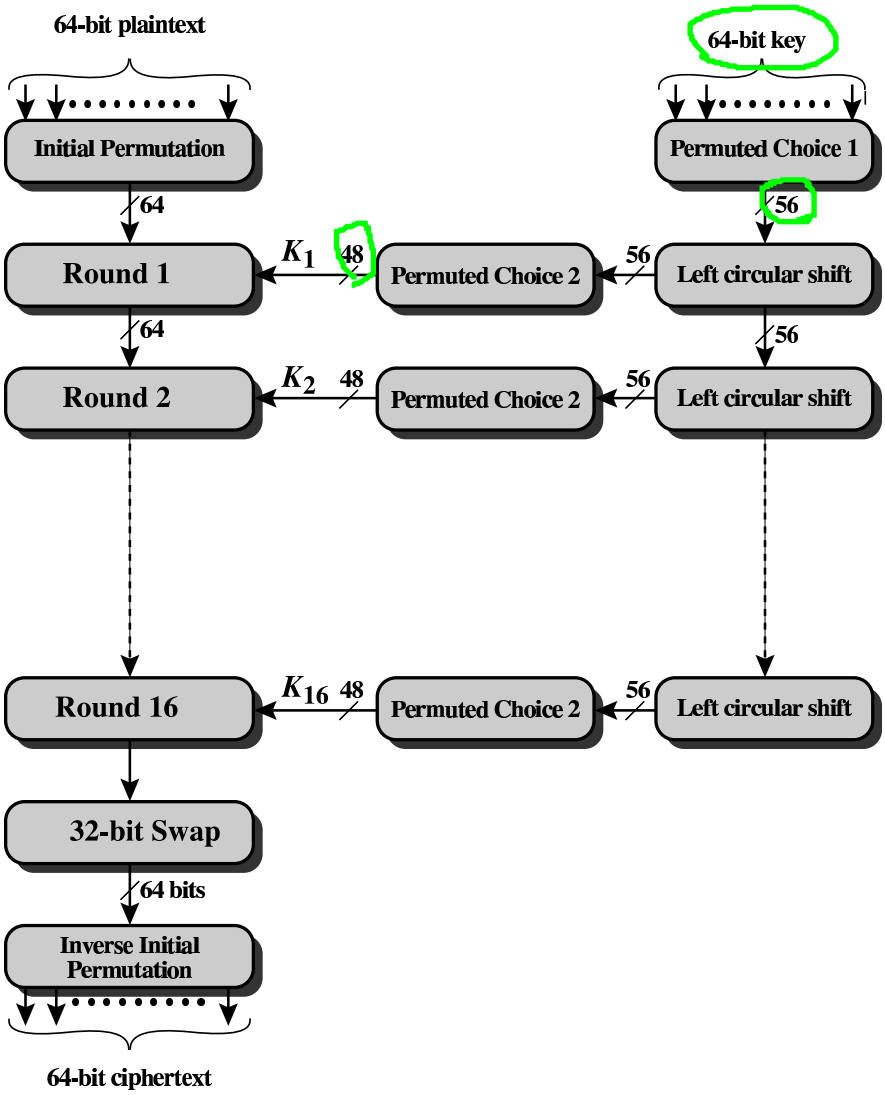

Figure 3.4 General Depiction of DES Encryption Algorithm

This figure is from Chapter 3 of Stallings: "Cryptography and Network Security", Fourth Edition

One Round of Processing in DEA

- The figure on the next page shows a single round of processing in DEA. The dotted rectangle constitutes the **F** function.

- The 32-bit right half of the 64-bit input data block is expanded by into a 48-bit block. This is accomplished by permutation and by repeating some of the bits. The conversion of the 32-bit right block into a 48-bit block is referred to as the **permutation/expansion** step.

- The 56-bit key is divided into two halves, each half shifted separately, and the combined 56-bit key **permuted/contracted** to yield a 48-bit **round** key.

- The goal of the substitution step implemented by the **S-box** is to introduce **diffusion** in the generation of the output from the input. *Diffusion means that each plaintext bit must affect as many ciphertext bits as possible.*

- The strategy used for creating the different round keys from the main key is meant to introduce **confusion** into the encryption process. *Confusion in this context means that the relationship between the encryption key and the ciphertext must be as complex as possible.*

- Diffusion and Confusion are the cornerstones of block cipher design.

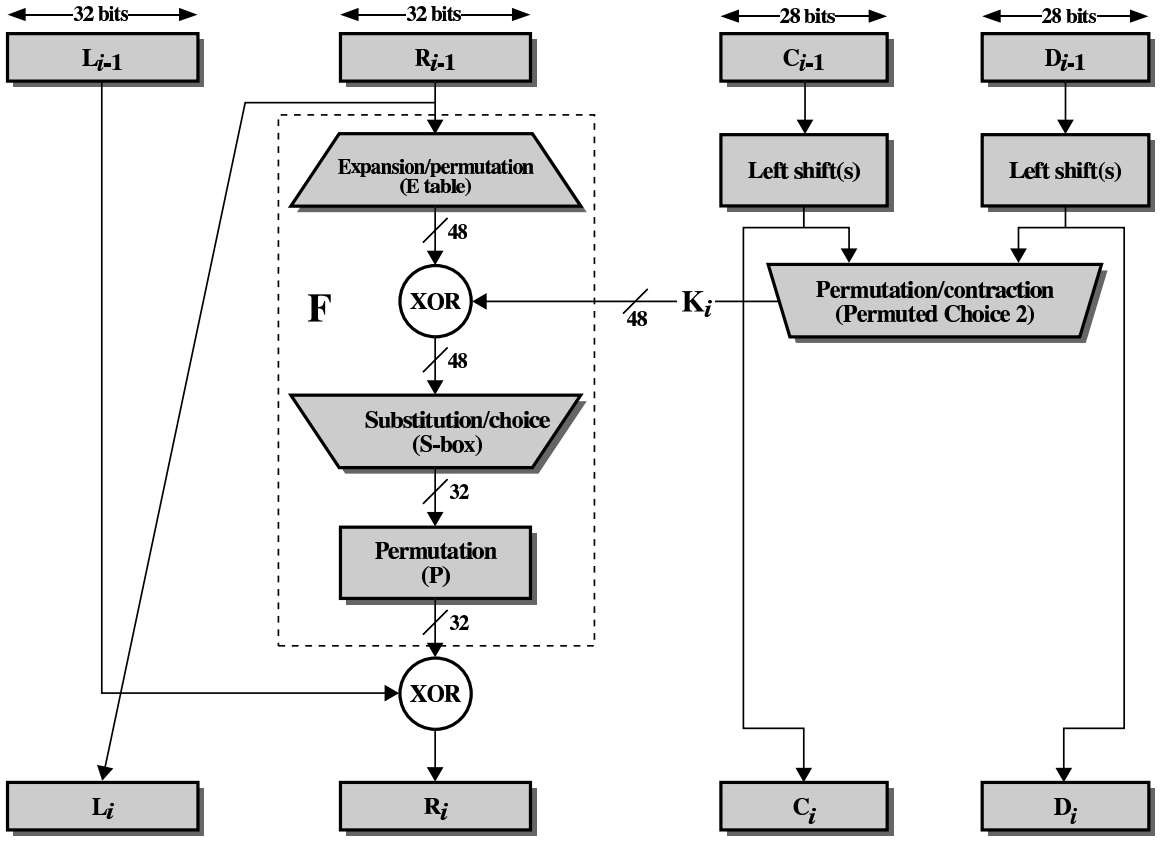

Figure 3.5 Single Round of DES Algorithm

This figure is from Chapter 3 of Stallings: "Cryptography and Network Security", Fourth Edition

The S-Box for the Substitution Step in Each Round

- The S-box step in the figure on the previous page is actually carried out by **eight** S-boxes. (See figure next page.)

- The 48-bit input word is divided into eight 6-bit words and each 6-bit word fed into a separate S-box. Each S-box produces a 4-bit output. Therefore, the 8 S-boxes together generate a 32-bit output. Therefore, the overall substitution step takes the 48-bit input back to a 32-bit output.

- Each of the eight S-boxes consists of a 4×16 table lookup for an output 4-bit word. The first and the last bit of the 6-bit input word are decoded into one of our rows and the middle 4 bits into one of 16 columns for the table lookup.

- To understand how the substitution step enhances diffusion, you have to understand the 32-bit to 48-bit expansion for the right block in the figure on the previous page. The expansion is carried out by
  - first dividing the 32-bit block into eight 4-bit words
  - attaching an additional bit on the left to each 4-bit word that is the last bit of the previous 4-bit word
  - attaching an additional bit to the right of each 4-bit word that is the beginning bit of the next 4-bit word.

- Thus, the row lookup for each of the eight S-boxes becomes a function of the input bits for the previous S-box and the next S-box.

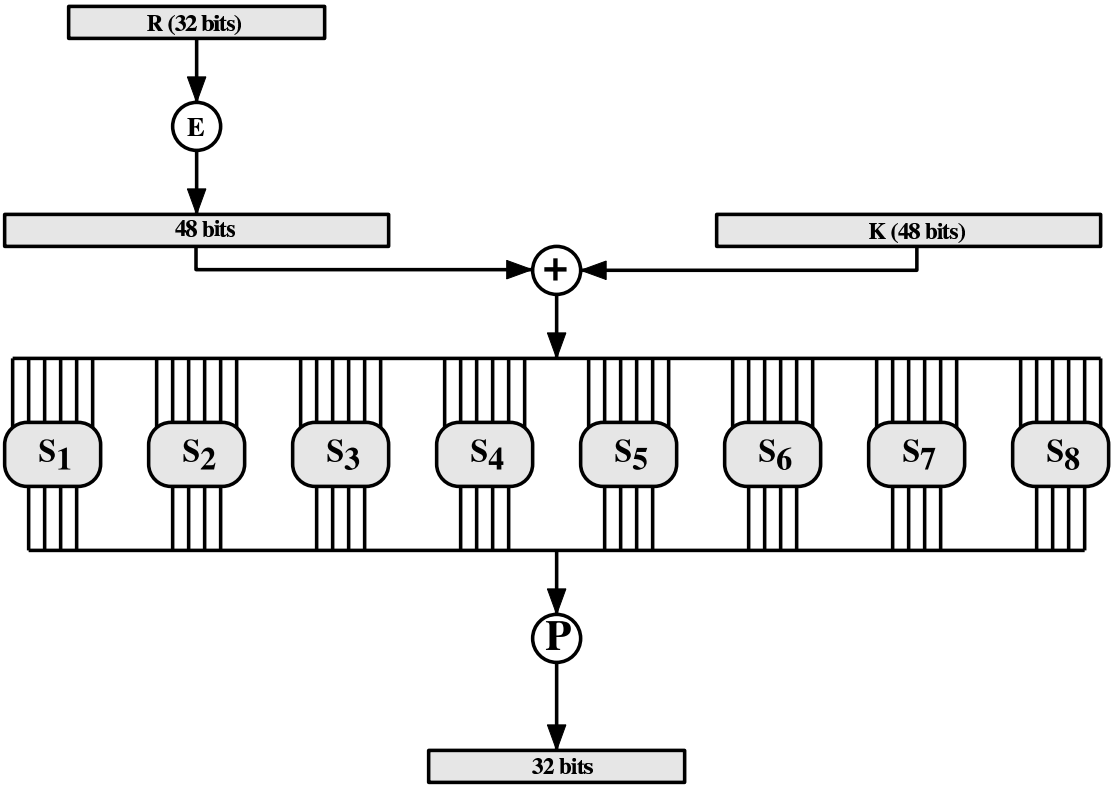

Figure 3.6 Calculation of F(R, K)

This figure is from Chapter 3 of Stallings: "Cryptography and Network Security", Fourth Edition

The Substitution Tables

The 4×16 substitution table for S_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

The 4×16 substitution table for S_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

One can similarly specify tables for the other fourteen substitution boxes.

As mentioned earlier, each row of a substitution table is indexed by the two outermost bits of a six-bit block and each column by the remaining inner 4 bit.

Round Key Generation

- The initial 56-bit key may be represented as 8 bytes, with the last bit of each byte used as a parity bit.

- The relevant 56 bits are subject to a permutation at the beginning before any round keys are generated. This is our Permutation Choice 1.

- At the beginning of each round, we divide the 56 relevant key bits into two 28 bit halves and circularly shift each half by one or two bits.

- For generating round key, we join together the two halves and apply a 56 bit to 48 bit contracting permutation (Permutation Choice 2) to the joined bit pattern. The resulting 48 bits constitute our round key.

- The two halves generated in each round are fed as the two halves going into the next round.

Initial Permutation of the Encryption Key

Permutation Choice 1						
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	32	44	36
63	35	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

8 rows

7 columns

Note that the bit positions assume that the key bits are addressed 1 through 64 in an 8-byte bit pattern. But note that the last bit of each byte is used as a parity bit. Also note that the permutation shown is not a table, in the sense that the rows and the columns do not carry any special and separate meanings. The permutation order for the bits is given by reading the entries shown from the upper left corner to the lower right corner.

Contraction-Permutation that Generates the 48-bit Round Key from the 56 Key

Permutation Choice 2							
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	32	31	37	47	55	30	40
51	45	33	48	44	49	39	56
34	53	46	42	50	36	29	32

6 rows

8 columns

As with the permutation shown on the previous page, note that the bit positions assume that the key bits are addressed 1 through 64 in an 8-byte bit pattern. But note that the last bit of each byte is used as a parity bit. Also note that the permutation shown is not a table, in the sense that the rows and the columns do not carry any special and separate meanings. The permutation order for the bits is given by reading the entries shown from the upper left corner to the lower right corner.

What Makes DES Strong (to the Extent it is Strong)

- The substitution step is very effective as far as **diffusion** is concerned. It has been shown that if you change just one bit of the 64-bit input data block, on the average that alters 34 bits of the ciphertext block.

- The manner in which the round keys are generated from the encryption key is also very effective as far as **confusion** is concerned. It has been shown that if you change just one bit of the encryption key, on the average that changes 35 bits of the ciphertext.

- Both effects mentioned above are referred to as the **avalanche effect**.

- And, of course, the 56-bit encryption key means a key space of size $2^{56} \approx 7.2 \times 10^{16}$. Assuming that, on the average, you'd need to try half the keys in a brute-force attack, a machine trying one key per microsecond would take 1142 years to break the code. However, a parallel-processing machine trying 1 million keys simultaneously would need only about 10 hours. (EFF took three days on a specially architected machine to break the code.)