SCSX1026 Cryptography and Network Security

UNIT II

UNIT II BLOCK CIPHERS AND THE DATA ENCRYPTION STANDARD (DES)

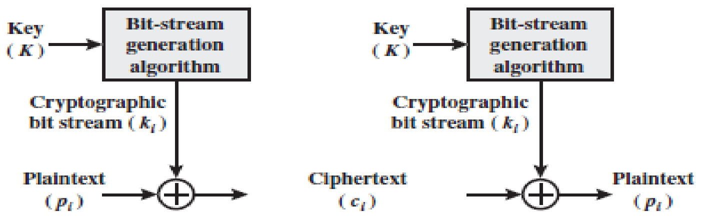
Simplified DES – Block Cipher principles – The Data Encryption Standard – The strength of DES – Confidentiality using symmetric encryption – Placement of encryption – Traffic confidentiality – Key distribution – Random number generation.

****

BLOCK CIPHER PRINCIPLES

Stream Ciphers and Block Ciphers

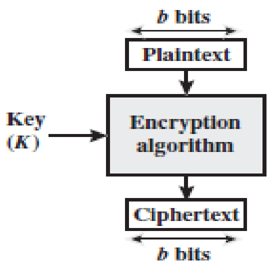
A **stream cipher** is one that encrypts a digital data stream one bit or one byte at a time. Examples of classical stream ciphers are the autokeyed Vigenère cipher and the Vernam cipher. Accordingly, for practical reasons, the bit-stream generator must be implemented as an algorithmic procedure, so that the cryptographic bit stream can be produced by both users.


Key (K)
Bit-stream generation algorithm
Cryptographic bit stream (k_i)
Plaintext (p_i)
Key (K)
Bit-stream generation algorithm
Cryptographic bit stream (k_i)
Ciphertext (c_i)
Plaintext (p_i)

(a) Stream cipher using algorithmic bit-stream generator

In this approach, the bit-stream generator is a key-controlled algorithm and must produce a bit stream that is cryptographically strong. Now, the two users need only share the generating key, and each can produce the keystream.

A **block cipher** is one in which a block of plaintext is treated as a whole and used to produce a ciphertext block of equal length. Typically, a block size of 64 or 128 bits is used. As with a stream cipher, the two users share a symmetric encryption key.

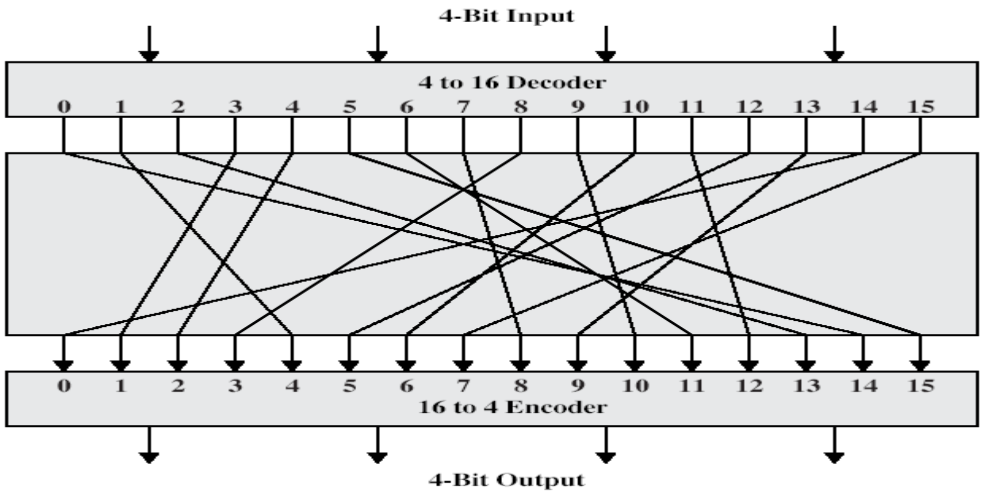


(b) Block cipher

A block cipher can be used to achieve the same effect as a stream cipher. In general, they seem applicable to a broader range of applications than stream ciphers. The vast majority of network-based symmetric cryptographic applications make use of block ciphers.

Motivation for the Feistel Cipher Structure

Encryption should be reversible. Figure shows the logic of a general substitution cipher for n=4 (block size).



General n-bit-n-bit Block Substitution (shown with $n = 4$)

In general the logic of a general substitution cipher for n=4 with 4-bit input produces one of 16 possible input states, which is mapped by the substitution cipher into a unique one of 16 possible output states, each of which is represented by 4 cipher text bits. The encryption and decryption mappings can be defined by tabulation, as shown below.

Encryption and Decryption Tables for Substitution

Plaintext	Ciphertext
0000	1110
0001	0100
0010	1101
0011	0001
0100	0010
0101	1111
0110	1011
0111	1000
1000	0011
1001	1010
1010	0110
1011	1100
1100	0101
1101	1001
1110	0000
1111	0111

Ciphertext	Plaintext
0000	1110
0001	0011
0010	0100
0011	1000
0100	0001
0101	1100
0110	1010
0111	1111
1000	0111
1001	1101
1010	1001
1011	0110
1100	1011
1101	0010
1110	0000
1111	0101

This is the most general form of block cipher and can be used to define any reversible mapping between plaintext and ciphertext. Feistel refers to this as the *ideal block cipher*, because it allows for the maximum number of possible encryption mappings from the plaintext block.

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 corresponding cipher text 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.

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

DIFFUSION AND CONFUSION

These are measures to thwart cryptanalysis based on statistical analysis. In diffusion, the statistical structure of the plaintext is dissipated into long range statistics of the ciphertext. This is achieved by having each plaintext letter affect the value of many ciphertext digits, which is equivalent to saying that each ciphertext digit is affected by many plaintext digits. An example of diffusion is to encrypt a message m=m1,m2,m3,.. Of characters with an averaging operation:

$$y_n = \sum_{i=1}^{k} m_{n+i} \pmod{26}$$

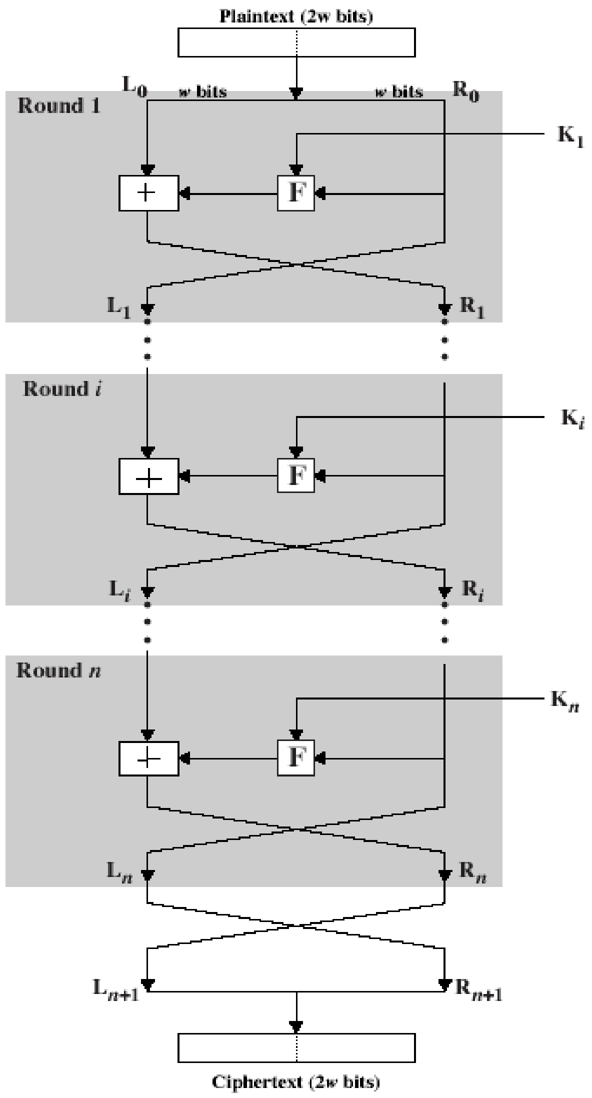
Adding k successive letters to get a ciphertext letter y_n. The letter frequencies in the ciphertext will be more nearly equal than in the plaintext (structure dissipated).

Confusion seeks to make the relationship between the statistics of the ciphertext and and the value of the encryption key as complex as possible. This is achieved by the use of a complex substitution algorithm. These operations became the cornerstone of modern block cipher design**.**

FEISTEL CIPHER STRUCTURE

The inputs to the encryption algorithm are a plaintext block of length 2w bits and a key K. The plaintext block is divided into 2 halves, L0 and R0. The 2 halves of the data pass through n rounds of processing and the combine to produce the ciphertext block. Each round i has as inputs L i-1 and Ri-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 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. Following this substitution, a 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.



Classical Feistel Network

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

Block size: large size means greater security but greater overhead (64, 128 bits)

Key size: large size means greater security but greater overhead (64, 128 bits)

Number of rounds: multiple rounds increase security (16 rounds)

Subkey generation algorithm: greater complexity – more secure

Round function: greater complexity – more secure

Additionally:

Fast software encryption/decryption: speed of execution becomes a concern

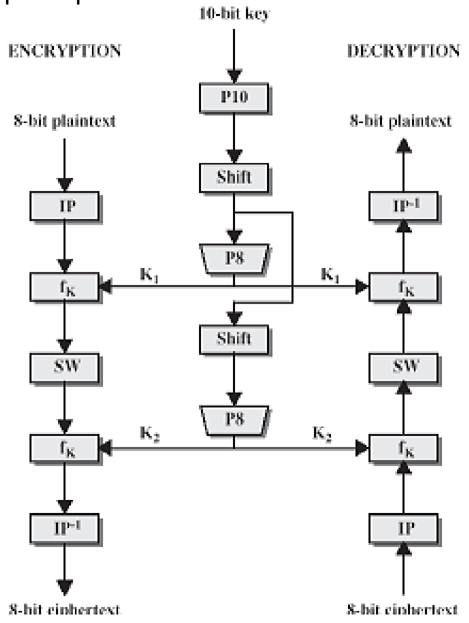
Ease of analysis: it should be difficult to cryptanalyze, but easy to analyze for cryptanalytic vulnerabilities.

We can see that SDES exhibits a Feistel structure with 2 rounds. The one difference from a "pure" Feistel structure is that the algorithm begins and ends with a permutation function. This difference also appears in full DES.

****

SIMPLIFIED DES

S-DES encryption algorithm takes 8-bit block of plaintext and a 10-bit key, and produces 8-bit ciphertext block. Encryption algorithm involves 5 functions: an initial permutation (IP); a complex function f_K, which involves both permutation and substitution and depends on a key input; a simple permutation function that switches (SW) the 2 halves of the data; the function f_K again; and finally, a permutation function that is the inverse of the initial permutation (IP-1). Decryption process is similar.



The function fK takes 8-bit key which is obtained from the 10-bit initial key. The key is first subjected to a permutation P10. Then a shift operation is performed. The output of the shift operation then passes through a permutation function that produces an 8-bit output (P8) for the first subkey (K1). The output of the shift operation also feeds into another shift and another instance of P8 to produce the 2nd subkey K2.

We can express encryption algorithm as superposition:

$$\text{Ciphertext} = IP^{-1}\,(f_{K_2}(SW(f_{K_1}(IP(plaintext)))))$$

Where,

$$K_1 = P8(Shift(P10(key)))$$

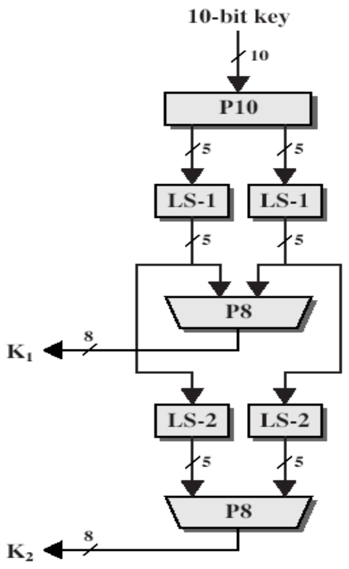
$$K_2 = P8(Shift(Shift(P10(key))))$$

Decryption is the reverse of encryption:

$$\text{Plaintext} = IP^{-1}\left(f_{K_1}(SW(f_{K_2}(IP(ciphertext))))\right)$$

We now examine S-DES in more details

Scheme of key generation



First, permute the 10-bit key k1,k2,......,k10:

P10(k1,k2,k3,k4,k5,k6,k7,k8,k9,k10) = (k3,k5,k2,k7,k4,k10,k1,k9,k8,k6)

Or it may be represented in such a form:

P10									
3	5	2	7	4	10	1	9	8	6

Each position in this table gives the identity of the input bit that produces the output bit in this position. So, the 1st output bit is bit 3 (k3), the 2nd is k5 and so on. For example, the key (1010000010) is permuted to (1000001100).

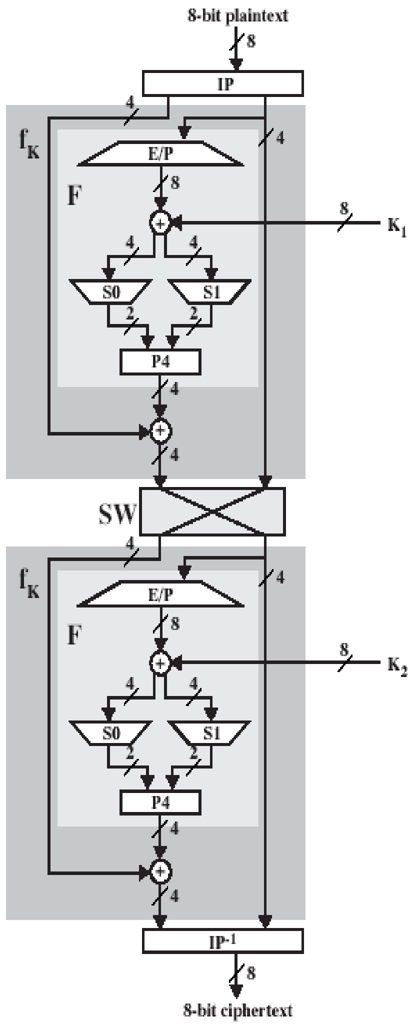
Next, perform a circular shift (LS-1), or rotation, separately on the 1st 5 bits and the 2nd 5 bits. In our example, the result is (00001 11000) Next, we apply P8, which picks out and permutes 8 out of 10 bits according to the following rule:

P8							
6	3	7	4	8	5	10	9

The result is subkey is K1. In our example, this yields (10100100)

We then go back to the pair of 5-bit strings produced by the 2 LS-1 functions and perform a circular left shift of 2 bit positions on each string. In our example, the value (00001 11000) becomes (00100 00011). Finally, P8 is applied again to produce K2. In our example, the result is (01000011).

S-DES ENCRYPTION



The input to the algorithm is an 8-bit block of plaintext, which is permuted by IP function:

IP							
2	6	3	1	4	8	5	7

At the end of the algorithm, the inverse permutation is used:

IP^{-1}							
4	1	3	5	7	2	8	6

It may be verified, that $IP^{-1}(IP(X)) = X$.

The most complex component of S-DES is the function f_K, which consists of a combination of permutation and substitution functions. The function can be expressed as follows. Let L and R be the leftmost 4 bits and rightmost 4 bits of the 8-bit input to f_K, and let F be a mapping (not necessarily one to one) from 4-bit strings to 4-bit strings. Then we let

$$f_K(L,R) = (L \oplus F(R,SK),R)$$

where SK is a subkey and ⊕ is the bit-by-bit XOR operation. For example, suppose the output of the IP stage in Fig.3.3 is (1011 1101) and F(1101,SK) = (1110) for some key SK. Then f_K(1011 1101) = (0101 1101) because (1011) ⊕ (1110) = (0101).

We now describe the mapping F. The input is a 4-bit number (n1 n2 n3 n4). The 1st operation is an expansion/permutation:

E/P							
4	1	2	3	2	3	4	1

For what follows, it is clearer to depict result in this fashion:

n4|n1 n2|n3
n2|n3 n4|n1

The 8-bit subkey K1 = (k11, k12, k13, k14, k15, k16, k17, k18) is added to this value using XOR:

n4+k11|n1+k12 n2+k13|n3+k14
n2+k15|n3+k16 n4+k17|n1+k18

Let us rename these bits:

p00|p01 p02|p03
p10|p11 p12|p13

The 1st 4 bits (1st row of the preceding matrix) are fed into the S-box S0 to produce a 2-bit output, and the remaining 4 bits (2nd row) are fed into S1 to produce another 2-bit output. These 2 boxes are defined as follows:

$$S0 = \begin{array}{c} 0\,1\,2\,3 \\ \begin{pmatrix} 1 & 0 & 3 & 2 \\ 3 & 2 & 1 & 0 \\ 0 & 2 & 1 & 3 \\ 3 & 1 & 3 & 2 \end{pmatrix} \end{array} \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \end{array} \quad S1 = \begin{array}{c} 0\,1\,2\,3 \\ \begin{pmatrix} 0 & 1 & 2 & 3 \\ 2 & 0 & 1 & 3 \\ 3 & 0 & 1 & 0 \\ 2 & 1 & 0 & 3 \end{pmatrix} \end{array} \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \end{array}$$

The S-boxes operate as follows. The 1st and 4th input bits are treated as a 2-bit number that specify a row of the S-box, and the 2nd and 3rd input bits specify a column of the S-box. The entry in that row and column, in base 2, is the 2-bit output. For example, if (p00, p03) = (00) and (p01, p02) = (10), then the output is from row 0, column 2 of S0, which is 3, or (11) in binary. Similarly, (p10, p13) and (p11, p12) are used to index into a row and column of S1 to produce an additional 2 bits.

Next, the 4 bits produced by S0 and S1 undergo a further permutation as follows:

P4			
2	4	3	1

The output of P4 is the output of function F.

The function f_K only alters the leftmost 4 bits of input. The switch function SW interchanges the left and right bits so that the 2nd instance of f_K operates on a different 4 bits. In the 2nd instance, the E/P, S0, S1, and P4 functions are the same. The key input is K2.

ANALYSIS OF SIMPLIFIED DES

A brute-force attack on S-DES is feasible since with a 10-bit key there are only 1024 possibilities.

What about cryptanalysis? If we know plaintext (p1 p2 p3 p4 p5 p6 p7 p8) and respective ciphertext (c1 c2 c3 c4 c5 c6 c7 c8), and key (k1 k2 k3 k4 k5 k6 k7 k8 k9 k10) is unknown, then we can express this problem as a system of 8 nonlinear equations with 10 unknowns. The nonlinearity comes from the S-boxes. It is useful to write down equations for these boxes. For clarity, rename (p00,p01,p02,p03)=(a,b,c,d) and (p10,p11,p12,p13)=(w,x,y,z). Then the operation of S0 is defined in the following equations:

$$q=abcd+ab+ac+b+d$$
$$r=abcd+abd+ab+ac+ad+a+c+1$$

where all additions are made modulo 2. Similar equations define S1. Alternating linear maps with these nonlinear maps results in very complex polynomial expressions for the cipher text bits, making cryptanalysis difficult.

****

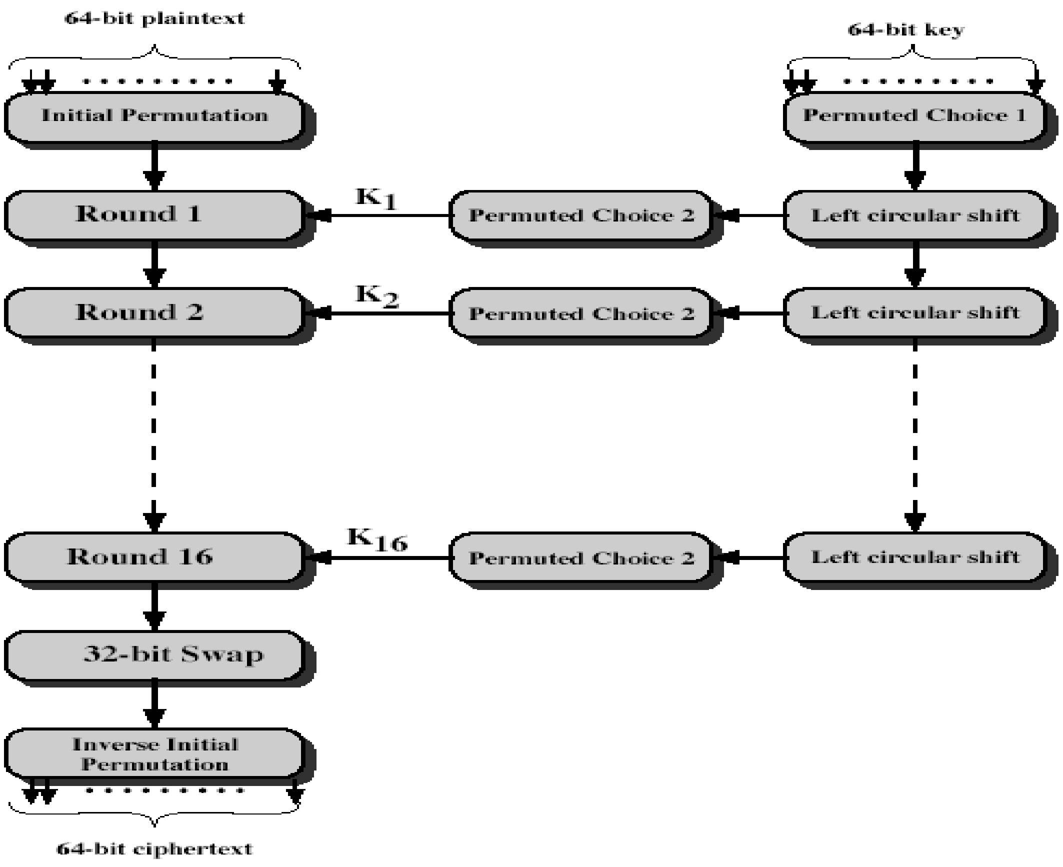
THE DATA ENCRYPTION STANDARD

DATA ENCRYPTION STANDARD

It was adopted in 1977 by the National Bureau of Standards (NBS), now National Institute of Standards and Technology (NIST), as Federal Information Processing Standard 46 (FIPS PUB 46). In 1971, IBM's team under Horst Feistel leadership developed algorithm LUCIFER, operating on 64-bit blocks with 128-bit key. Further, IBM's team leaded by Walter Tuchman and Carl Meyer revised LUCIFER to make it more resistant to cryptanalysis, but they reduced key size to 56 bits. In 1973, NBS issued a request for proposals for a national cipher standard. IBM submitted results of its Tuchman-Meyer project. This was by far the best algorithm proposed and was adopted in 1977 as Data Encryption Standard. In 1994, NIST reaffirmed DES for federal use for another 5 years. In 1999, NIST issued a new version of its standard (FIPS PUB 46-3) that indicated that DES should only be used for legacy systems and that triple DES be used.

The Data Encryption Standard (DES) is a symmetric-key block cipher published by the National Institute of Standards and Technology (NIST).

DES is an implementation of a Feistel Cipher. It uses 16 round Feistel structure. The block size is 64-bit. Though, key length is 64-bit, DES has an effective key length of 56 bits, since 8 of the 64 bits of the key are not used by the encryption algorithm (function as check bits only).



General Depiction of DES Encryption Algorithm

The 64 bit input enters into initial permutation and the permutated output is fed into sixteen rounds with kay values and then 32-bit swap swaps left and 32-bit halves obtained after Round 16, we get preoutput. Finally, preoutput passes through a permutation IP-1, that is inverse to initial permutation IP, to produce the 64-bit ciphertext. The right-hand portion of Fig. 3.7 shows the way in which 56-bit is used. For each of 16 rounds a subkey Ki is produced by the combination of a left circular shift and a permutation. The permutation function is the same for each round**.**

INITIAL PERMUTATION AND ITS INVERSE

It effects on 64-bit input :

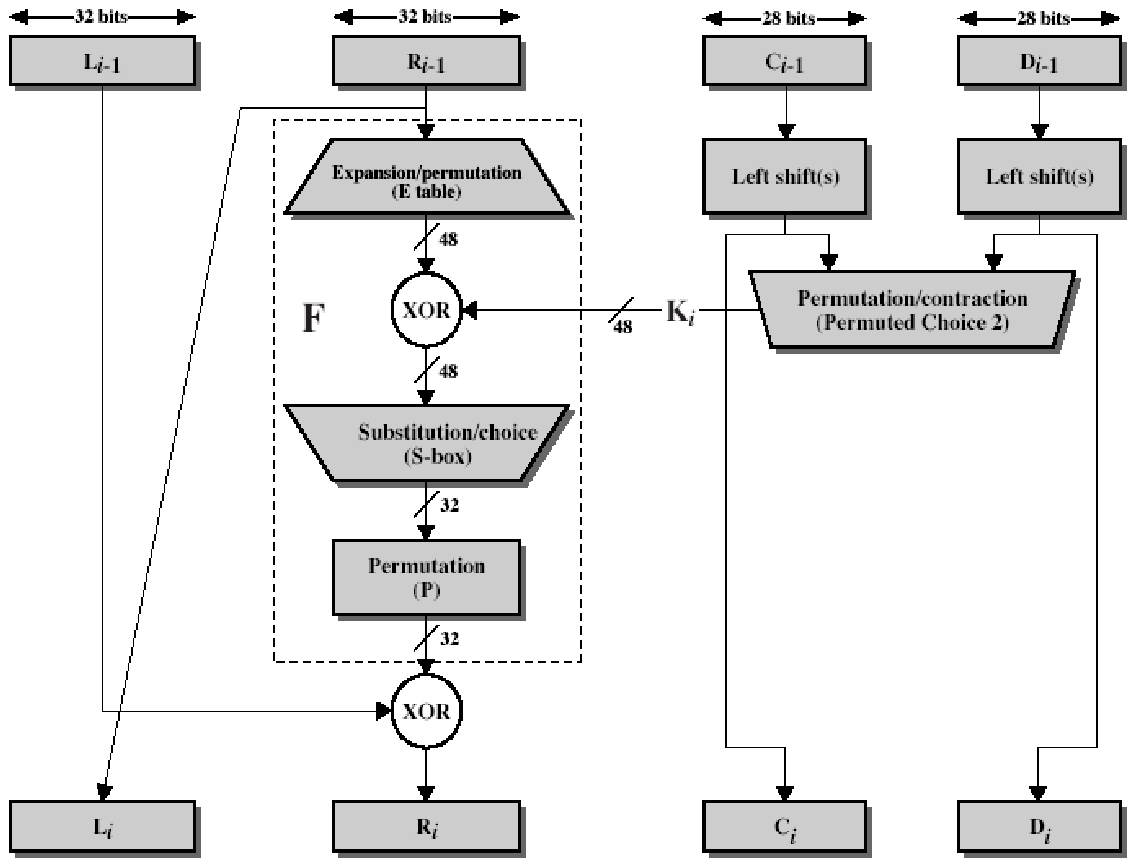
IP:

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

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

DETAILS OF SINGLE ROUND



Single Round of DES Algorithm

The left and right halves of each 64-bit intermediate value are treated as separate 32-bit quantities, labeled L and R. As in the classic Feistel cipher, the overall process at each round is summarized as follows:

$$L_i = R_{i-1}$$
$$R_i = L_{i-1} \oplus F(R_{i-1}, K_i)$$

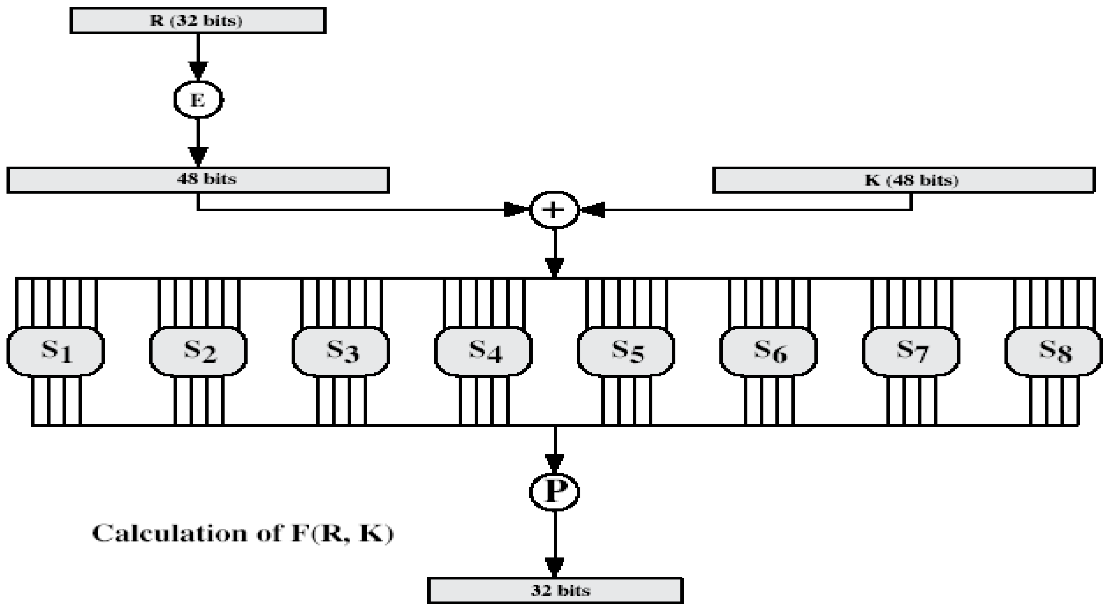
The round key Ki is 48 bits. The R input is 32 bits. This R input is first expanded to 48 bits by Expansion/Permutation (E table):

Expansion/Permutation (E / P table)					
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

The resulting 48 bits are XORed with Ki. This 48 bit result passes through a substitution function that produces 32-bit output, which is permuted by Permutation function (P):

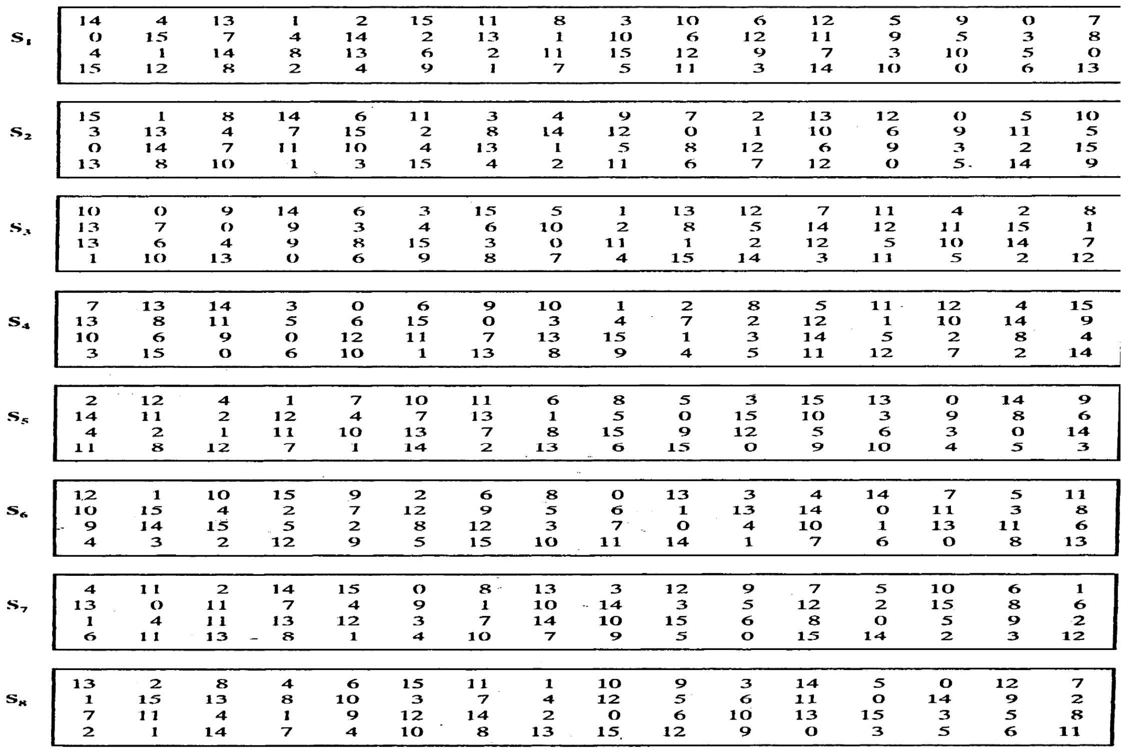
Permutation function(P)							
16	7	20	21	29	12	28	17
1	15	23	26	5	18	31	10
2	8	24	14	32	27	3	9
19	13	30	6	22	11	4	25

The role of S-boxes is illustrated in the below figure:



The substitution consists of a set of 8 S-boxes, each of which accepts 6 bits input and produces 4 bits as output. These transformations are:

Each row of an S-box defines a general reversible substitution: middle 4 bits of each group of 6-bit input are substituted by S-box output, 1st and last 6th bits define what particular substitution out of to use.



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
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
S_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_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_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_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_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_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

KEY GENERATION

Input key has 64 bits. But each 8th bit is not used: bits 8,16,24,32,40,48,56,64 are not further used. The 56-bit key is first subjected to permutation Permuted Choice 1:

Permuted Choice 1 (PC-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 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

The resulting 56-bit key is then treated as 2 28-bit quantities, labeled 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 the following:

Schedule of Left Shifts																
Round number	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Bits rotated	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

These shifted values serve as input to the next round. They also serve as input to Permuted Choice 2, which produces a 48-bit output that serves as input to the function

$F(R_{i-1}, K_i)$.

Permuted Choice 2 (PC-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	52	31	37	47	55	30	40
51	45	33	48	44	49	39	56
34	53	46	42	50	36	29	32

DES DECRYPTION

As with any Feistel cipher, decryption uses the same algorithm as encryption, except that the application of subkeys is reversed.

THE AVALANCE EFFECT IN DES

Avalanche effect − A small change in plaintext results in the very grate change in the ciphertext. 1 bit change in the plaintext leads to 34 bit difference in the ciphertext. 1 bit change in the key leads to 35 bit difference in the ciphertext.

****

THE STRENGTH OF DES

DES proved insecure in July 1998, when the Electronic Frontier Foundation (EFF) announced that it had broken a DES encryption using a special-purpose “DES cracker” machine that was built for less than $250 000. The attack took less than 3 days.

Design criteria for S-boxes were not made public, so there was a concern that cryptanalysis is possible for an opponent who knows the weaknesses in S-boxes. Up to now, there are no published results about such weaknesses in S-boxes.

DES also appears to be resistant to timing attack but suggest some avenues to explore. Timing attack tries to understand essence of algorithm by analysis of time of its work on different inputs.

One of such approaches yields a Hamming weight (number of bits equal to 1) of the secret key.

****

CONFIDENTIALITY USING SYMMETRIC ENCRYPTION

We begin with a discussion of the location of encryption logic; the main choice here is between what are known as link encryption and end-to-end encryption.

Next, we look at the use of encryption to counter traffic analysis attacks. Then we discuss the difficult problem of key distribution. Finally, we discuss the principles underlying an important tool in providing a confidentiality facility: random number generation.

PLACEI

If encryp what to
encrypt section
examine ɔ major
approac

There ar for wide
area cor the end
user. Ev ures are
possible

Link ver

The mos ty
highligh counter
these at ion gear
should b **d-to-end**
encrypti

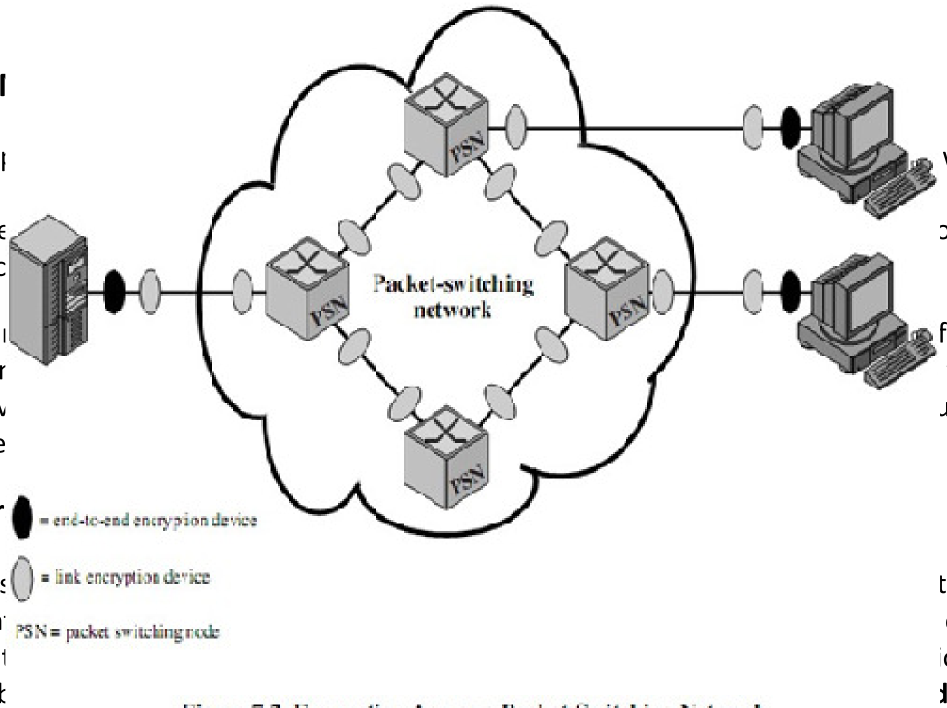


Figure 7.2 Encryption Across a Packet-Switching Network

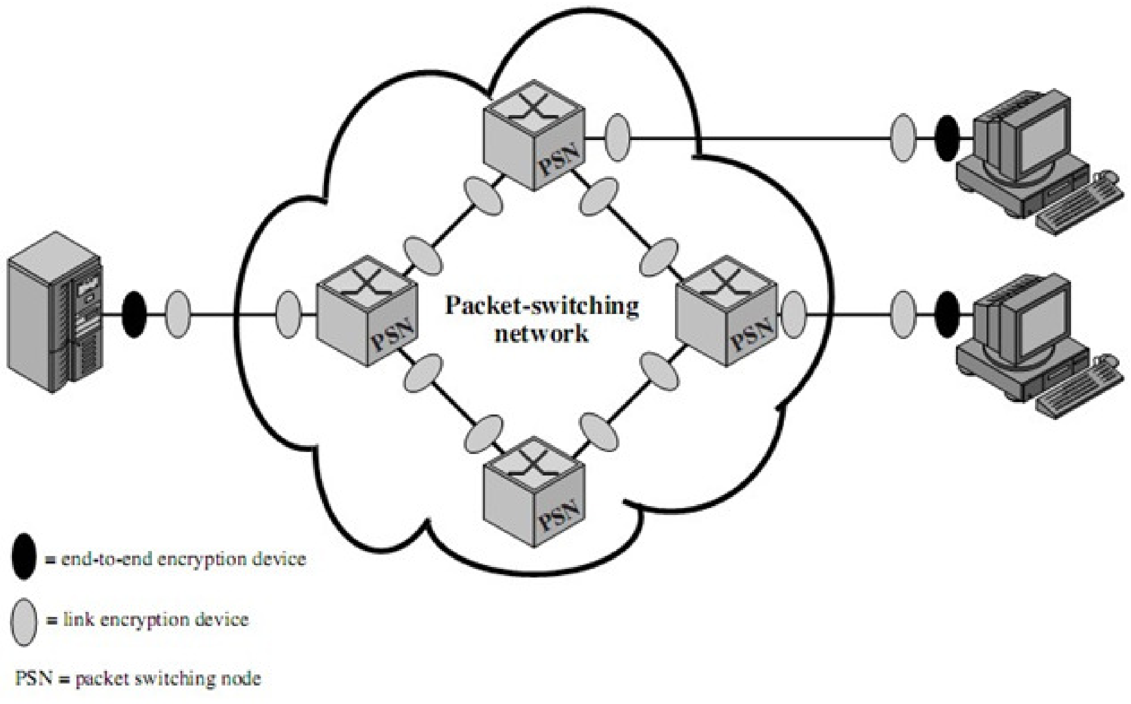


Figure 7.2 Encryption Across a Packet-Switching Network

Link to Link Encryption:

With link encryption, each vulnerable communications link is equipped on both ends with an encryption device. Thus, all traffic over all communications links is secured. One of its disadvantages is that the message must be decrypted each time it enters a switch because the switch must read the address (logical connection number) in the packet header in order to route the frame. Thus, the message is vulnerable at each switch. If working with a public network, the user has no control over the security of the nodes.

Several implications of link encryption should be noted. For this strategy to be effective, all the potential links in a path from source to destination must use link encryption. Each pair of nodes that share a link **should share a unique key, with a different key used on each link.** Thus, many keys must be provided.

End-To-End Encryption

With end-to-end encryption, the encryption process is carried out at the two end systems. The source host or terminal encrypts the data. The data in encrypted form are then transmitted unaltered across the network to the destination terminal or host. The destination shares a key with the source and so is able to decrypt the data. This plan seems to secure the transmission against attacks on the network links or switches. Thus, end-to-end encryption relieves the end user of concerns about the degree of security of networks and links that support the communication. There is, however, still a weak spot.

Consider the following situation. A host connects to a frame relay or ATM network, sets up a logical connection to another host, and is prepared to transfer data to that other host by using end-to-end encryption. Data are transmitted over such a network in the form of packets that consist of a header and some user data. What part of each packet will the host encrypt? Suppose that the host encrypts the entire packet, including the header. This will not work because, remember, only the other host can perform the decryption. The frame relay or ATM switch will receive an encrypted packet and be unable to read the header. Therefore, it will not be able to route the packet. It follows that the host may encrypt only the user data portion of the packet and must leave the header in the clear.

Thus, with end-to-end encryption, the user data are secure. However, the traffic pattern is not, because packet headers are transmitted in the clear. On the other hand, end-to-end encryption does provide a degree of authentication. If two end systems share an encryption key, then a recipient is assured that any message that it receives comes from the alleged sender, because only that sender shares the relevant key. Such authentication is not inherent in a link encryption scheme.

To achieve greater security, both link and end-to-end encryptions are needed, as is shown in Figure. When both forms of encryption are employed, the host encrypts the user data portion of a packet using an end-to-end encryption key. The entire packet is then encrypted using a link encryption key. As the packet traverses the network, each switch decrypts the packet, using a link encryption key to read the header, and then encrypts the entire packet again for sending it out on the next link. Now the entire packet is secure except for the time that the packet is actually in the memory of a packet switch, at which time the packet header is in the clear.

Link Encryption	End-to-End Encryption
Link encryption encrypts all the data along a specific communication path. Not only is the user information encrypted, but the header, trailers, addresses, and routing data that are part of the packets are also encrypted.	End-to-end encryption, the headers, addresses, routing, and trailer information are not encrypted, enabling attackers to learn more about a captured packet and where it is headed.
All data are encrypted, including headers, addresses, and routing information.	Headers, addresses, and routing information are not encrypted, and therefore not protected.
It works at a lower layer in the OSI model.	It works at Network layer.
All of the information is encrypted, and the packets must be decrypted at each hop so the router, or other intermediate device, knows where to send the packet next.	The packets do not need to be decrypted and then encrypted again at each hop, because the headers and trailers are not encrypted.
Requires one key per host pair. Provides Host Authentication	Requires one key per user pair. Provides user Authentication

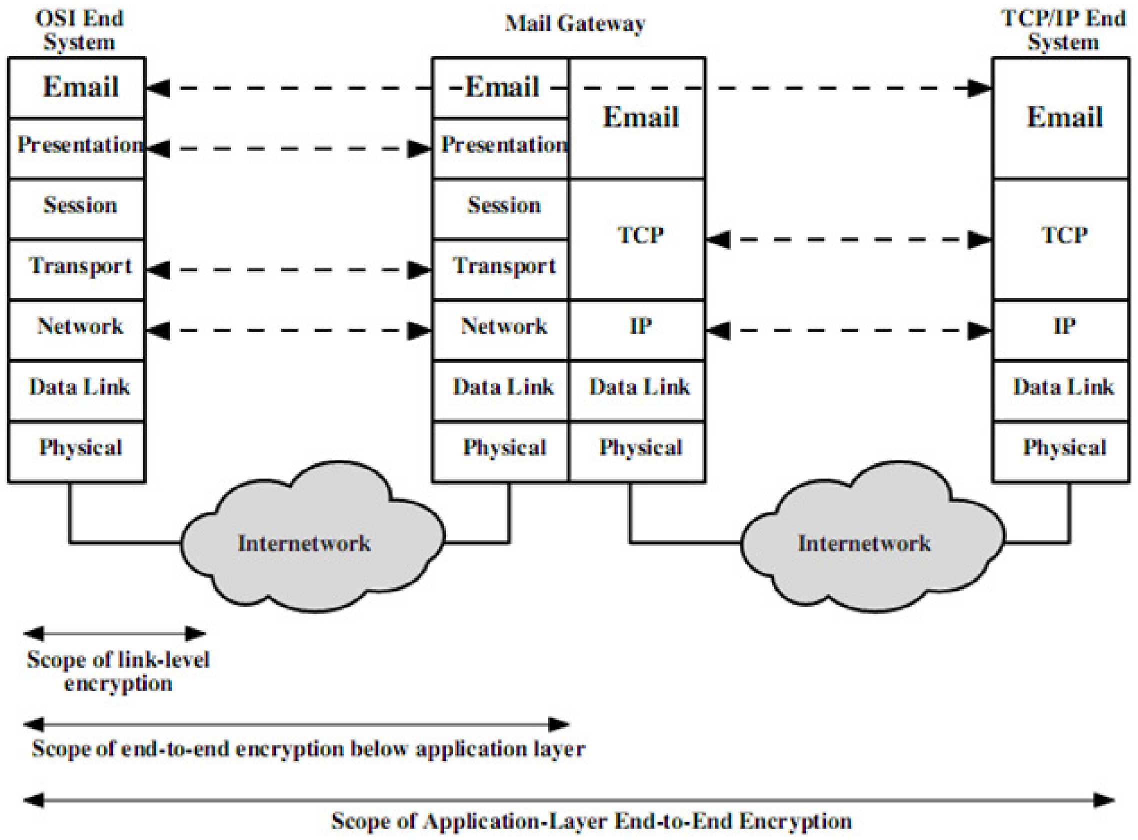
Logical Placement of End-to-End Encryption Function

With link encryption, the encryption function is performed at a low level of the communications hierarchy i.e. physical or link layers.

For end-to-end encryption, several choices are possible for the logical placement of the encryption function. At the lowest practical level, the encryption function could be performed at the network layer.

With network-layer encryption, Each end system can engage in an encrypted exchange with another end system if the two share a secret key. All the user processes and applications within each end system would employ the same encryption scheme with the same key to reach a particular target end system.

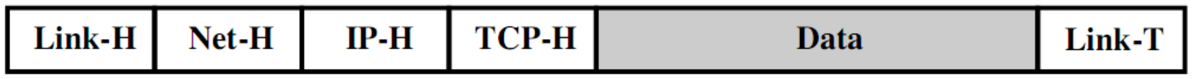
The following figure illustrates the issues involved. In this example, an electronic mail gateway is used to interconnect an internetwork that uses a TCP/IP-based architecture. In such a configuration, there is no end-to-end protocol below the application layer. The transport and network connections from each end system terminate at the mail gateway, which sets up new transport and network connections to link to the other end system. Even if both end systems use TCP/IP or OSI, there are plenty of instances in actual configurations in which mail gateways sit between otherwise isolated internetworks. Thus, for applications like electronic mail that have a store-and-forward capability, the only place to achieve end-to-end encryption is at the application layer.


OSI End System
Mail Gateway
TCP/IP End System
Email
Presentation
Session
Transport
Network
Data Link
Physical
TCP
IP
Internetwork
Scope of link-level encryption
Scope of end-to-end encryption below application layer
Scope of Application-Layer End-to-End Encryption

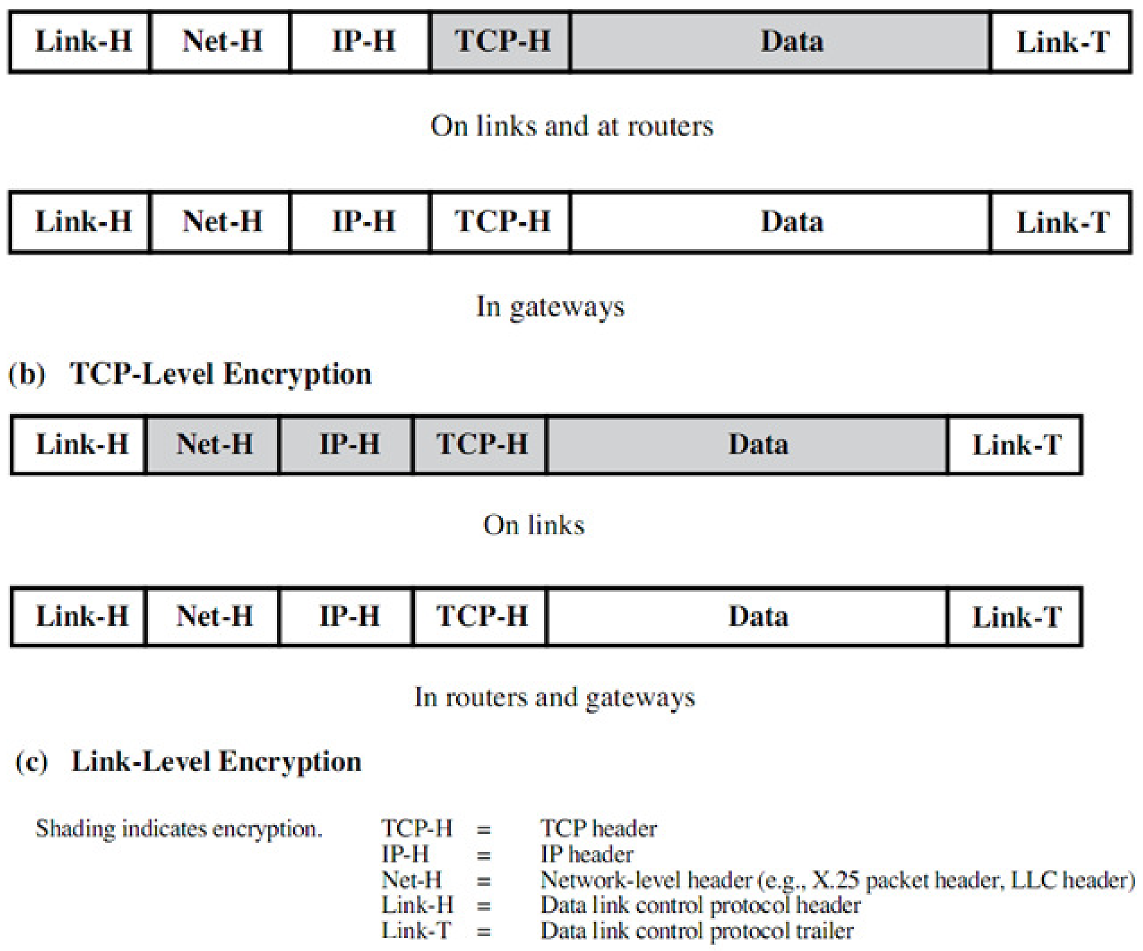
Encryption Coverage Implications of Store-and-Forward Communications

A drawback of application-layer encryption is that the number of entities to consider increases dramatically. A network that supports hundreds of hosts may support thousands of users and processes. Thus, many more secret keys need to be generated and distributed.

An interesting way of viewing the alternatives is to note that as we move up the communications hierarchy, less information is encrypted but it is more secure.



(a) Application-Level Encryption (on links and at routers and gateways)


Link-H
Net-H
IP-H
TCP-H
Data
Link-T
On links and at routers
In gateways
(b) TCP-Level Encryption
On links
In routers and gateways
(c) Link-Level Encryption
Shading indicates encryption.
TCP-H = TCP header
IP-H = IP header
Net-H = Network-level header (e.g., X.25 packet header, LLC header)
Link-H = Data link control protocol header
Link-T = Data link control protocol trailer

Relationship between Encryption and Protocol Levels

With application-level encryption as shown in above figure, only the user data portion of a TCP segment is encrypted. The TCP, IP, network-level, and link-level headers and link-level trailer are in the clear. By contrast, if encryption is performed at the TCP level , then, on a single end-to-end connection, the user data and the TCP header are encrypted. The IP header remains in the clear because it is needed by routers to route the IP datagram from source to destination. Note, however, that if a message passes through a gateway, the TCP connection is terminated and a new transport connection is opened for the next hop.

Furthermore, the gateway is treated as a destination by the underlying IP. Thus, the encrypted portions of the data unit are decrypted at the gateway. If the next hop is over a TCP/IP network, then the user data and TCP header are encrypted again before transmission. However, in the gateway itself the data unit is buffered entirely in the clear. Finally, for link-level encryption , the entire data unit except for the link header and trailer is encrypted on each link, but the entire data unit is in the clear at each router and gateway.

****

TRAFFIC CONFIDENTIALITY

The following types of information that can be derived from a traffic analysis attack:

- Identities of partners
- How frequently the partners are communicating
- Message pattern, message length, or quantity of messages that suggest important information is being exchanged
- The events that correlate with special conversations between particular partners

Another concern related to traffic is the use of traffic patterns to create a **covert channel**. Typically, the channel is used to transfer information in a way that violates a security policy. For example, an employee may wish to communicate information to an outsider in a way that is not detected by management and that requires simple eavesdropping on the part of the outsider.

Link Encryption Approach

With the use of link encryption, network-layer headers (e.g., frame or cell header) are encrypted, reducing the opportunity for traffic analysis. However, it is still possible in those circumstances for an attacker to assess the amount of traffic on a network and to observe the amount of traffic entering and leaving each end system. An effective countermeasure to this attack is traffic padding.

Traffic padding produces cipher text output continuously, even in the absence of plaintext. A continuous random data stream is generated. When plaintext is available, it is encrypted and transmitted. When input plaintext is not present, random data are encrypted and transmitted. This makes it impossible for an attacker to distinguish between true data flow and padding and therefore impossible to deduce the amount of traffic.

End-to-End Encryption Approach

Traffic padding is essentially a link encryption function. If only end-to-end encryption is employed, then the measures available to the defender are more limited. For example, if encryption is implemented at the application layer, then an opponent can determine which transport entities are engaged in dialogue.

One technique that might prove useful is to pad out data units to a uniform length at either the transport or application level. In addition, null messages can be inserted randomly into the stream. These tactics deny opponent knowledge about the amount of data exchanged between end users and obscure the underlying traffic pattern.

****

KEY DISTRIBUTION

For symmetric encryption to work, the two parties to an exchange must share the same key, and that key must be protected from access by others. Furthermore, frequent key changes are usually desirable to limit the amount of data compromised if an attacker learns the key. Therefore, the term that refers to the means of delivering a key to two parties who wish to exchange data, without allowing others to see the key. For two parties A and B, key distribution can be achieved in a number of ways, as follows:

1. A can select a key and physically deliver it to B.
2. A third party can select the key and physically deliver it to A and B.
3. If A and B have previously and recently used a key, one party can transmit the new key to the other, encrypted using the old key.
4. If A and B each has an encrypted connection to a third party C, C can deliver a key on the encrypted links to A and B.

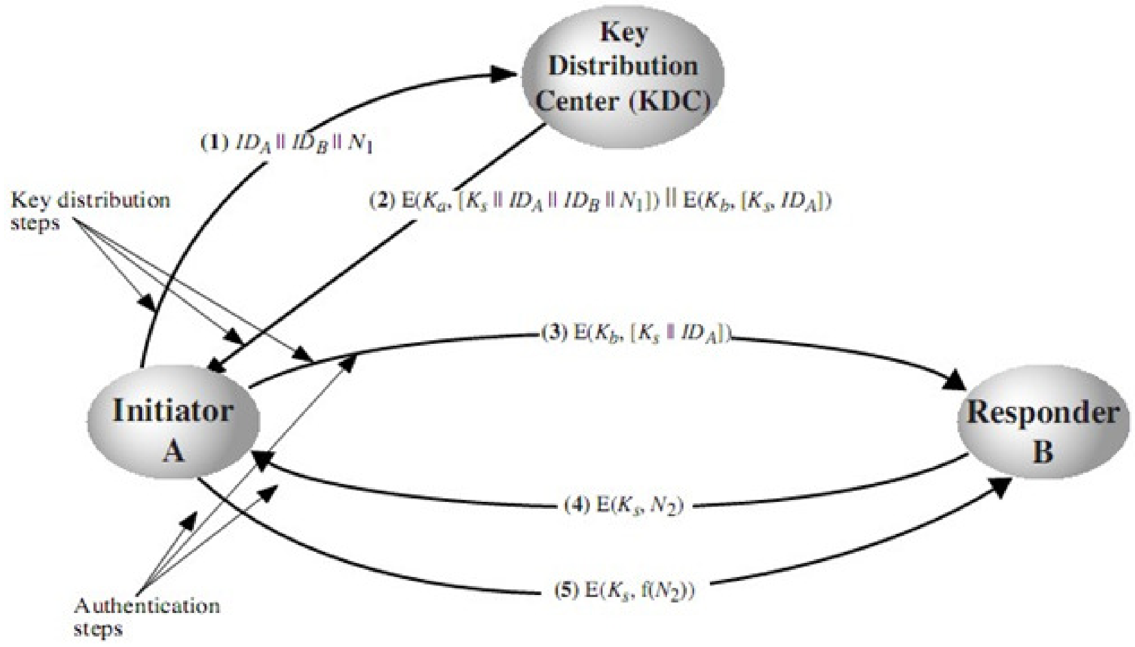
Physical delivery (1 & 2) is simplest - but only applicable when there is personal contact between recipient and key issuer. This is fine for link encryption where devices & keys occur in pairs, but does not scale as number of parties who wish to communicate grows. 3 is mostly based on 1 or 2 occurring first.

A third party, whom all parties trust, can be used as a **trusted intermediary** to mediate the establishment of secure communications between them (4). Must trust intermediary not to abuse the knowledge of all session keys. As number of parties grow, some variant of 4 is only practical solution to the huge growth in number of keys potentially needed.

Key distribution centre:

- The use of a key distribution center is based on the use of a hierarchy of keys. At a minimum, two levels of keys are used.
- Communication between end systems is encrypted using a temporary key, often referred to as a **session key**.
- Typically, the session key is used for the duration of a logical connection and then discarded
- **master key** is shared by the key distribution center and an end system or user and used to encrypt the session key.

Key Distribution Scenario:


Key Distribution Center (KDC)
(1) $ID_A \| ID_B \| N_1$
Key distribution steps
(2) $E(K_a, [K_s \| ID_A \| ID_B \| N_1]) \| E(K_b, [K_s, ID_A])$
(3) $E(K_b, [K_s \| ID_A])$
Initiator A
Responder B
(4) $E(K_s, N_2)$
(5) $E(K_s, f(N_2))$
Authentication steps

Key Distribution Scenario

Let us assume that user A wishes to establish a logical connection with B and requires a one-time session key to protect the data transmitted over the connection. A has a master key, K_a, known only to itself and the KDC; similarly, B shares the master key K_b with the KDC. The following steps occur:

1. A issues a request to the KDC for a session key to protect a logical connection to B. The message includes the identity of A and B and a unique identifier, N_1, for this transaction, which we refer to as a **nonce**. The nonce may be a timestamp, a counter, or a random number; the minimum requirement is that it differs with each request. Also, to prevent masquerade, it should be difficult for an opponent to guess the nonce. Thus, a random number is a good choice for a nonce.

2. The KDC responds with a message encrypted using Ka Thus, A is the only one who can successfully read the message, and A knows that it originated at the KDC. The message includes two items intended for A:

   - The one-time session key, Ks, to be used for the session
   - The original request message, including the nonce, to enable A to match this response with the appropriate request

   Thus, A can verify that its original request was not altered before reception by the KDC and, because of the nonce, that this is not a replay of some previous request.

   In addition, the message includes two items intended for B:

   - The one-time session key, Ks to be used for the session
   - An identifier of A (e.g., its network address), IDA

These last two items are encrypted with Kb (the master key that the KDC shares with B). They are to be sent to B to establish the connection and prove A's identity.

3. A stores the session key for use in the upcoming session and forwards to B the information that originated at the KDC for B, namely, $E(K_b, [K_s || ID_A])$. Because this information is encrypted with K_b, it is protected from eavesdropping. B now knows the session key (K_s), knows that the other party is A (from ID_A), and knows that the information originated at the KDC (because it is encrypted using K_b).

   At this point, a session key has been securely delivered to A and B, and they may begin their protected exchange. However, two additional steps are desirable:

4. Using the newly minted session key for encryption, B sends a nonce, N_2, to A.

5. Also using K_s, A responds with $f(N_2)$, where f is a function that performs some transformation on N_2 (e.g., adding one).

These steps assure B that the original message it received (step 3) was not a replay.

Note that the actual key distribution involves only steps 1 through 3 but that steps 4 and 5, as well as 3, perform an authentication function.

Major Issues with KDC:

For very large networks, a hierarchy of KDCs can be established. For communication among entities within the same local domain, the local KDC is responsible for key distribution. If two entities in different domains desire a shared key, then the corresponding local KDCs can communicate through a (hierarchy of) global KDC(s)

To balance security & effort, a new session key should be used for each new connection-oriented session. For a connectionless protocol, a new session key is used for a certain fixed period only or for a certain number of transactions.

An automated key distribution approach provides the flexibility and dynamic characteristics needed to allow a number of terminal users to access a number of hosts and for the hosts to exchange data with each other, provided they trust the system to act on their behalf.

The use of a key distribution center imposes the requirement that the KDC be trusted and be protected from subversion. This requirement can be avoided if key distribution is fully decentralized. In addition to separating master keys from session keys, may wish to define different types of session keys on the basis of use.

****

RANDOM NUMBER GENERATION

Random Numbers many uses of random numbers in cryptography

– nonces in authentication protocols to prevent replay

– session keys

– public key generation

– keystream for a one-time pad

• in all cases its critical that these values be

– Statistically random, uniform distribution, independent

– Unpredictability of future values from previous values

• true random numbers provide this

• care needed with generated random numbers

****