CMSC 487/687

Simplified Data Encryption Standard

This is a simplified version of the Data Encryption Standard. It is meant as a teaching tool to understand how the Data Encryption Standard operates. It is defined similarly to the original Data Encryption Standard, but uses much smaller key and block sizes and only consists of two rounds. Also included is the specification for the Double Simplified Data Encryption Algorithm, which will be used to demonstrate the Meet in the Middle attack.

1. **Name of Standard.** Simplified Data Encryption Standard (S-DES).
2. **Category of Standard.** Computer Security, Cryptography.
3. **Explanation.** The Simplified Data Encryption Standard (S-DES) specifies two cryptographic algorithms. When used in conjunction with American National Standards Institute (ANSI) X9.52 standard, this publication provides a complete description of the mathematical algorithms for encrypting (enciphering) and decrypting (deciphering) binary coded information. Encrypting data converts it to an unintelligible form called cipher. Decrypting cipher converts the data back to its original form called plaintext. The algorithms described in this standard specifies both enciphering and deciphering operations which are based on a binary number called a key.

A S-DES key consists of 10 binary digits (“0”s or “1”s) which are randomly generated and used directly by the algorithm. A DS-DEA key consists of two S-DES keys, which is also referred to as a key bundle. Authorized users of encrypted computer data must have the key that was used to encipher the data in order to decrypt it. The encryption algorithms specified in this standard are commonly known among those using the standard. The cryptographic security of the data depends on the security provided for the key used to encipher and decipher the data.

Data can be recovered from cipher only by using exactly the same key used to encipher it. Unauthorized recipients of the cipher who know the algorithm but do not have the correct key cannot derive the original data algorithmically. However, it may be feasible to determine the key by a brute force “exhaustion attack.” Also, anyone who does have the key and the algorithm can easily decipher the cipher and obtain the original data. A standard algorithm based on a secure key thus provides a basis for exchanging encrypted computer data by issuing the key used to encipher it to those authorized to have the data.

CMSC 487/687

February 22, 2019

SPECIFICATIONS FOR THE

SIMPLIFIED DATA ENCRYPTION STANDARD (S-DES)

The Simplified Data Encryption Standard (S-DES) shall consist of the following Data Encryption Algorithm (S-DES) and Double Simplified Data Encryption Algorithm DS-DEA. These devices shall be designed in such a way that they may be used in a computer system or network to provide cryptographic protection to binary coded data. The method of implementation will depend on the application and environment. The devices shall be implemented in such a way that they may be tested and validated as accurately performing the transformations specified in the following algorithms.

DATA ENCRYPTION ALGORITHM

Introduction

The algorithm is designed to encipher and decipher blocks of data consisting of 8 bits under control of a 10-bit key[1]. Deciphering must be accomplished by using the same key as for enciphering, but with the schedule of addressing the key bits altered so that the deciphering process is the reverse of the enciphering process. A block to be enciphered is subjected to an initial permutation ***IP***, then to a complex key-dependent computation and finally to a permutation which is the inverse of the initial permutation $\boldsymbol{IP^{-1}}$. The key-dependent computation can be simply defined in terms of a function ***f***, called the cipher function, and a function ***KS***, called the key schedule. A description of the computation is given first, along with details as to how the algorithm is used for encipherment. Next, the use of the algorithm for decipherment is described. Finally, a definition of the cipher function ***f*** is given in terms of primitive functions which are called the selection functions $\boldsymbol{S_i}$ and the permutation function ***P***. $\boldsymbol{S_i}$, ***P*** and ***KS*** of the algorithm are contained in Appendix 1.

[1] Blocks are composed of bits numbered from left to right, i.e., the left most bit of a block is bit one.

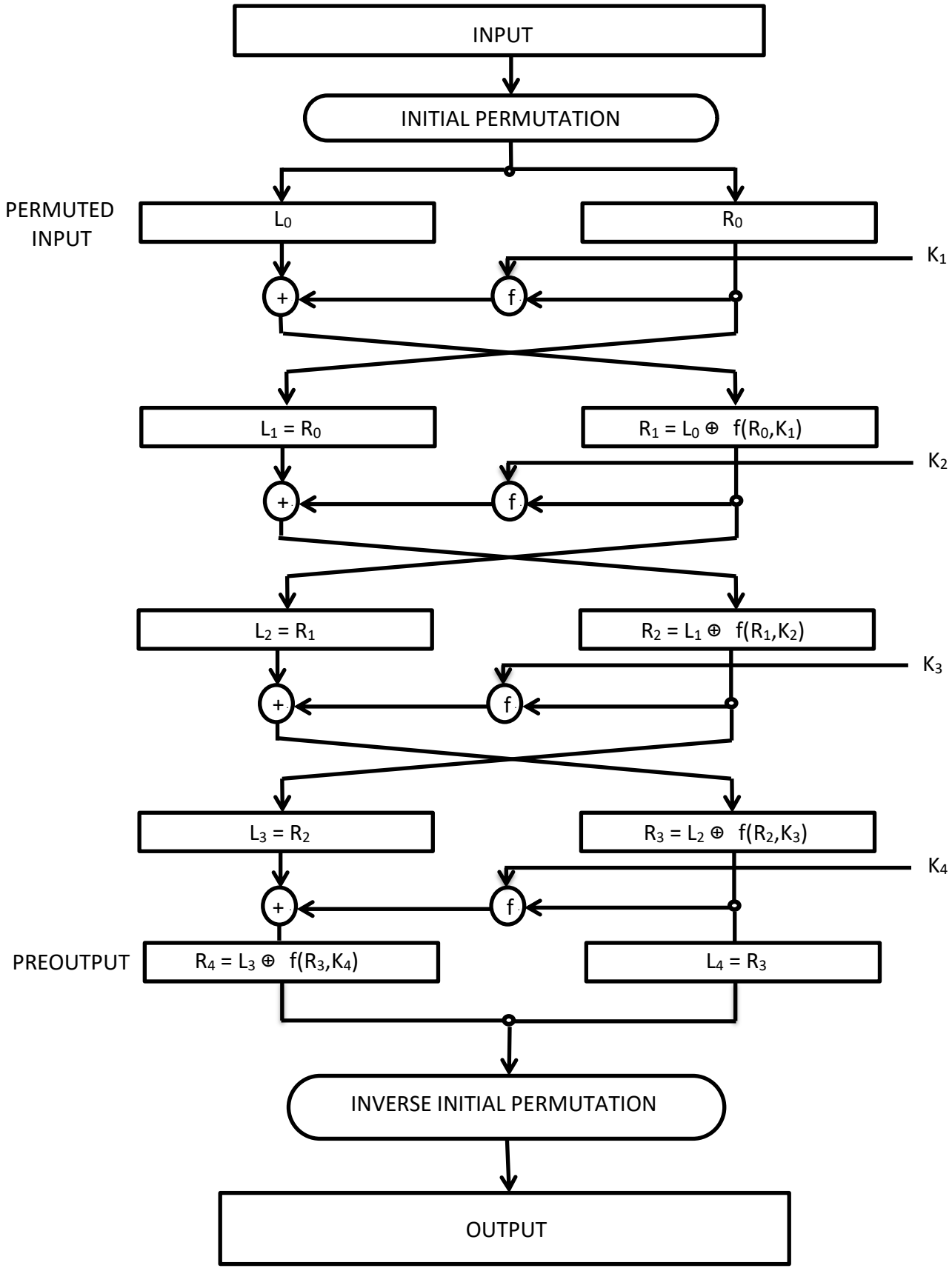


Figure 1. ***Enciphering computation.***

The following notation is convenient: Given two blocks ***L*** and ***R*** of bits, ***LR*** denotes the block consisting of the bits of ***L*** followed by the bits of ***R***. Since concatenation is associative, $B_1B_2...B_8$, for example, denotes the block consisting of the bits of B_1 followed by the bits of B_2…followed by the bits of B_8.

Enciphering

A sketch of the enciphering computation is given in **Figure 1.**

The 8 bits of the input block to be enciphered are first subjected to the following permutation, called the initial permutation ***IP***:

IP

2 6 3 1 4 8 5 7

That is the permuted input has bit 2 in its first bit, bit 6 as its second bit, and so on with bit 7 as its last bit. The permuted input block is then the input to a complex key-dependent computation described below. The output of that computation, called the preoutput, is then subjected to the following permutation which is the inverse of the initial permutation:

IP^{-1}

4 1 3 5 7 2 8 6

That is, the output of the algorithm has bit 4 of the preoutput block as its first bit, bit 1 as its second bit, and so on, until bit 6 of the preoutput block is the last bit of the output.

The computation which uses the permuted input block as its input to produce the preoutput block consists, but for a final interchange of blocks, of 2 iterations of a calculation that is described below in terms of the cipher function ***f*** which operates on two blocks, one of 4 bits and one of 8 bits, and produces a block of 4 bits.

Let the 8 bits of the input block to an iteration consist of a 4 bit block ***L*** followed by a 4 bit block ***R***. Using the notation defined in the introduction, the input block is then ***LR***.

Let ***K*** be a block of 8 bits chosen from the 10-bit key. Then the output ***L'R'*** of an iteration with input ***LR*** is defined by:

(1)
$$L' = R$$
$$R' = L \oplus f(R,K)$$

where $\oplus$ denotes bit-by-bit addition modulo 2.

As remarked before, the input of the first iteration of the calculation is the permuted input block. If $L'R'$ is the output of the 4th iteration then $R'L'$ is the preoutput block. At each iteration, a different block K of key bits is chosen from the 8-bit key designated by KEY.

With more notation, we can describe the iterations of the computation in more detail. Let KS be a function that takes an integer n in the range from 1 to 4 and a 10-bit block KEY as input and yields as output a 8-bit block K_n which is a permuted selection of bits from KEY. That is

(2) $$K_n = KS(n, KEY)$$

with K_n determined by the bits in 8 distinct bit positions of KEY. KS is called the key schedule because the block K used in the n'th iteration of (1) is the block K_n determined by (2).

As before, let the permuted input block be LR. Finally, let L_0 and R_0 be respectively L and R and let L_n and R_n be respectively L' and R' of (1) when L and R are respectively L_{n-1} and R_{n-1} and K is K_n; that is, when n is in the range from 1 to 4,

(3) $$L_n = R_{n-1}$$
$$R_n = L_{n-1} \oplus f(R_{n-1},K_n)$$

The preoutput block is then R_4L_4.

The key schedule KS of the algorithm is described in detail in the Appendix. The key schedule produces the 4 K_n, which are required for the algorithm.

Deciphering

The permutation IP^{-1} applied to the preoutput block is the inverse of the initial permutation IP applied to the input. Further, from (1) it follows that:

(4) $$R = L'$$
$$L = R' \oplus f(L',K)$$

Consequently, to ***decipher*** it is only necessary to apply the ***very same algorithm to an enciphered message block***, taking care that at each iteration of the computation ***the same block of key bits*** **K** ***is used*** during decipherment as was used during the encipherment of the block. Using the notation of the previous section, this can be expressed by the equations:

(5) $$R_{n-1} = L_n$$
$$L_{n-1} = R_n \oplus f(L_n,K_n)$$

where now R_4L_4 is the permuted input block for the deciphering calculation and L_0R_0 is the preoutput block. That is, for the decipherment calculation with R_4L_4 as the permuted input, K_4 is used in the first iteration and K_3 in the second and so forth.

The Cipher Function f

A sketch of the calculation of ***f(R,K)*** is given in **Figure 2**.

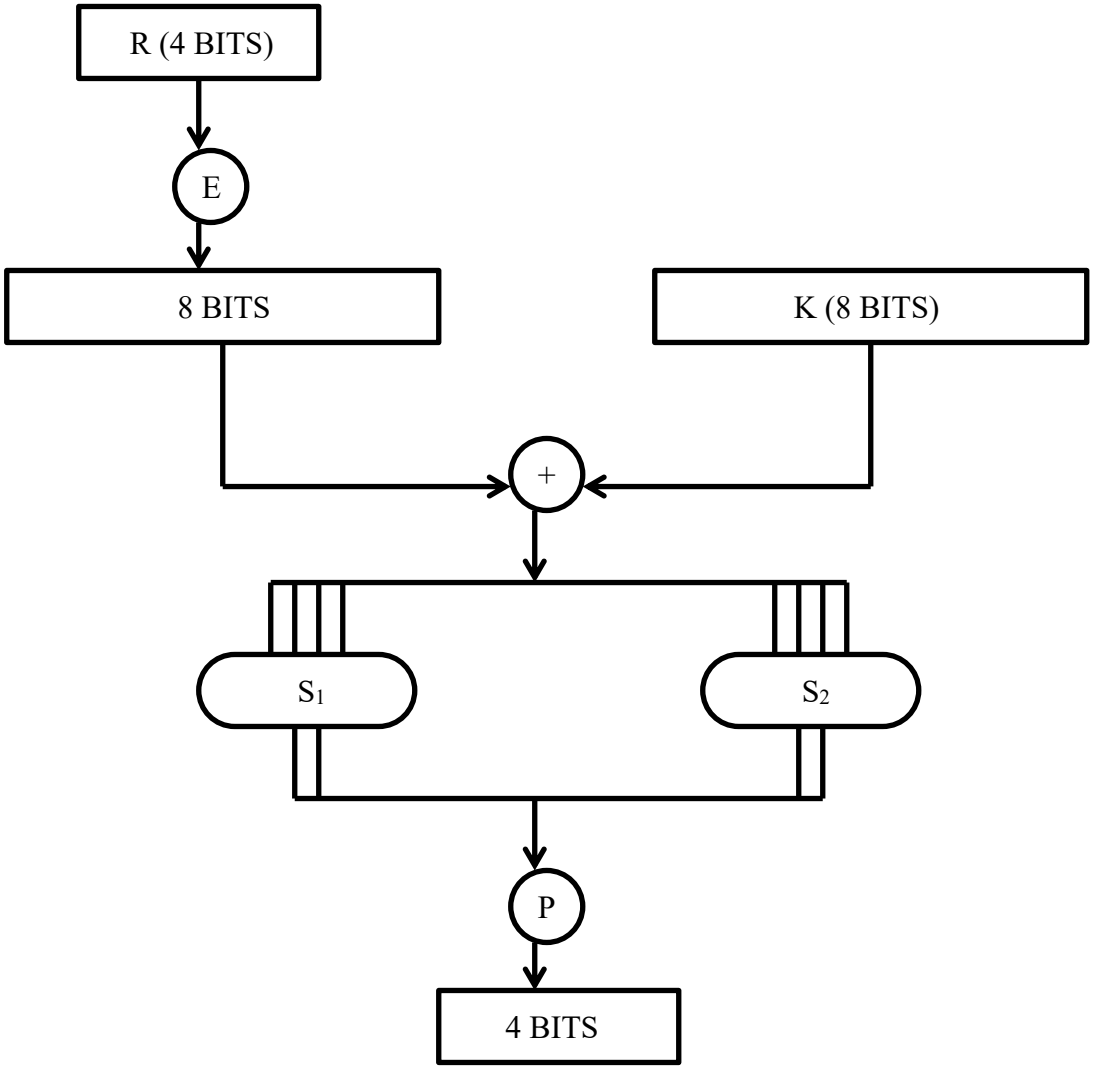


Figure 2. ***Calculation of f*****(R, K)**

Let ***E*** denote a function which takes a block of 4 bits as input and yields a block of 8 bits as output. Let ***E*** be such that the 8 bits of its output, written as 2 blocks of 4 bits each, are obtained by selecting the bits in its inputs in order according to the following table:

E **BIT-SELECTION TABLE**

4 1 2 3 2 3 4 1

Thus the first three bits of ***E(R)*** are the bits in positions 4, 1 and 2 of ***R*** while the last 2 bits of ***E(R)*** are the bits in positions 4 and 1.

Each of the unique selection functions S_1 and S_2, takes a 4-bit block as input and yields a 2-bit block as output and is illustrated by using a table containing the recommended S_1:

S_1

Column Number

Row	0	1	2	3
0	1	0	3	2
1	3	2	1	0
2	0	2	1	3
3	3	1	3	2

If S_1 is the function defined in this table and ***B*** is a block of 4 bits, then $S_1(B)$ is determined as follows: The first and last bits of ***B*** represent in base 2 a number in the range 0 to 3. Let that number be ***i***. The middle 2 bits of ***B*** represent in base 2 a number in the range 0 to 3. Let that number be ***j***. Look up in the table the number in the ***i***'th row and ***j***'th column. It is a number in the range 0 to 3 and is uniquely represented by a 2 bit block. That block is the output $S_1(B)$ of S_1 for the input ***B***. For example, for input 0110 the row is 00, that is row 0, and the column is determined by 11, that is column 3. In row 0 column 3 appears 2 so that the output is 10. Selection functions S_1 and S_2 of the algorithm appear in Appendix 1.

The permutation function ***P*** yields a 4-bit output from a 4-bit input by permuting the bits of the input block. Such a function is defined by the following table:

P

2 4 3 1

The permutation function P of the algorithm is repeated in Appendix 1.

Now let S_1 and S_2 be two distinct selection functions, let ***P*** be the permutation function and let ***E*** be the function defined above.

To define ***f(R,K)*** we first define B_1 and B_2 to be blocks of 4 bits each for which

(6) $$B_1B_2 = K \oplus E(R)$$

The block $f(R,K)$ is then defined to be

(7) $$P(S_1(B_1)S_2(B_2))$$

Thus $K \oplus E(R)$ is first divided into the 2 blocks as indicated in (6). Then each B_i is taken as an input to S_i and the 2 blocks $S_1(B_1)$ and $S_2(B_2)$ of 2 bits each are consolidated into a single block of 4 bits which forms the input to P. The output (7) is then the output of the function f for the inputs R and K.

DOUBLE SIMPLE DATA ENCRYPTION ALGORITHM

Let $E_K(I)$ represent the S-DES encryption of I using S-DES key K. Each DS-DEA encryption/decryption operation is a compound operation of S-DES encryption and decryption operations. The following operations are used:

1. DS-DEA encryption operations: the transformation of an 8-bit block I into an 8-bit block O that is defined as follows:
$$O = E_{K2}(E_{K1}(I)).$$

2. DS-DEA decryption operations: the transformation of an 8-bit block I into an 8-bit block O that is defined as follows:
$$O = D_{K1}(D_{K2}(I)).$$

The standard specifies that K_1 and K_2 are independent keys.

APPENDIX 1

PRIMITIVE FUNCTIONS FOR THE SIMPLIFIED DATA ENCRYPTION ALGORITHM

The choice of the primitive functions ***KS, S_1, S_2*** and ***P*** is critical to the strength of an encipherment resulting from the algorithm. Specified below is the recommended set of functions, describing ***S_1, S_2*** and ***P*** in the same way they are described in the algorithm. For the interpretation of the tables describing these functions, see the discussion in the body of the algorithm.

The primitive functions ***S_1*** and ***S_2*** are:

S_1

1	0	3	2
3	2	1	0
0	2	1	3
3	1	3	2

S_2

0	1	2	3
2	0	1	3
3	0	1	0
2	1	0	3

The primitive function ***P*** is:

2 4 3 1

Recall that K_n, for $1 \leq n \leq 4$, is the block of 8 bits in (2) of the algorithm. Hence, to describe ***KS***, it is sufficient to describe the calculation of K_n from KEY for $n = 1, 2, 3, 4$. That calculation is illustrated in **Figure 3**. To complete the definition of ***KS*** it is therefore sufficient to describe the two permuted choices, as well as the schedule of left shifts.

Permuted choice 1 is determined by the following table:

PC-1

3 5 2 7 4

10 1 9 8 6

The table has been divided into two parts, with the first part determining how the bits of $C_{(\ )}$ are chosen, and the second part determining how the bits of $D_{(\ )}$ are chosen. The bits

of ***KEY*** are numbered 1 through 10. The bits of C_0 are respectively bits 3, 5, 2, 7, and 4 of ***KEY***, with the bits of D_0 being bits 10, 1, 9, 8 and 6 of ***KEY***.

With C_0 and D_0 defined, we now define how the blocks C_n and D_n are obtained from the blocks C_{n-1} and D_{n-1}, respectively, for n = 1, 2, 3, 4. That is accomplished by adhering to the following schedule of left shifts of the individual blocks:

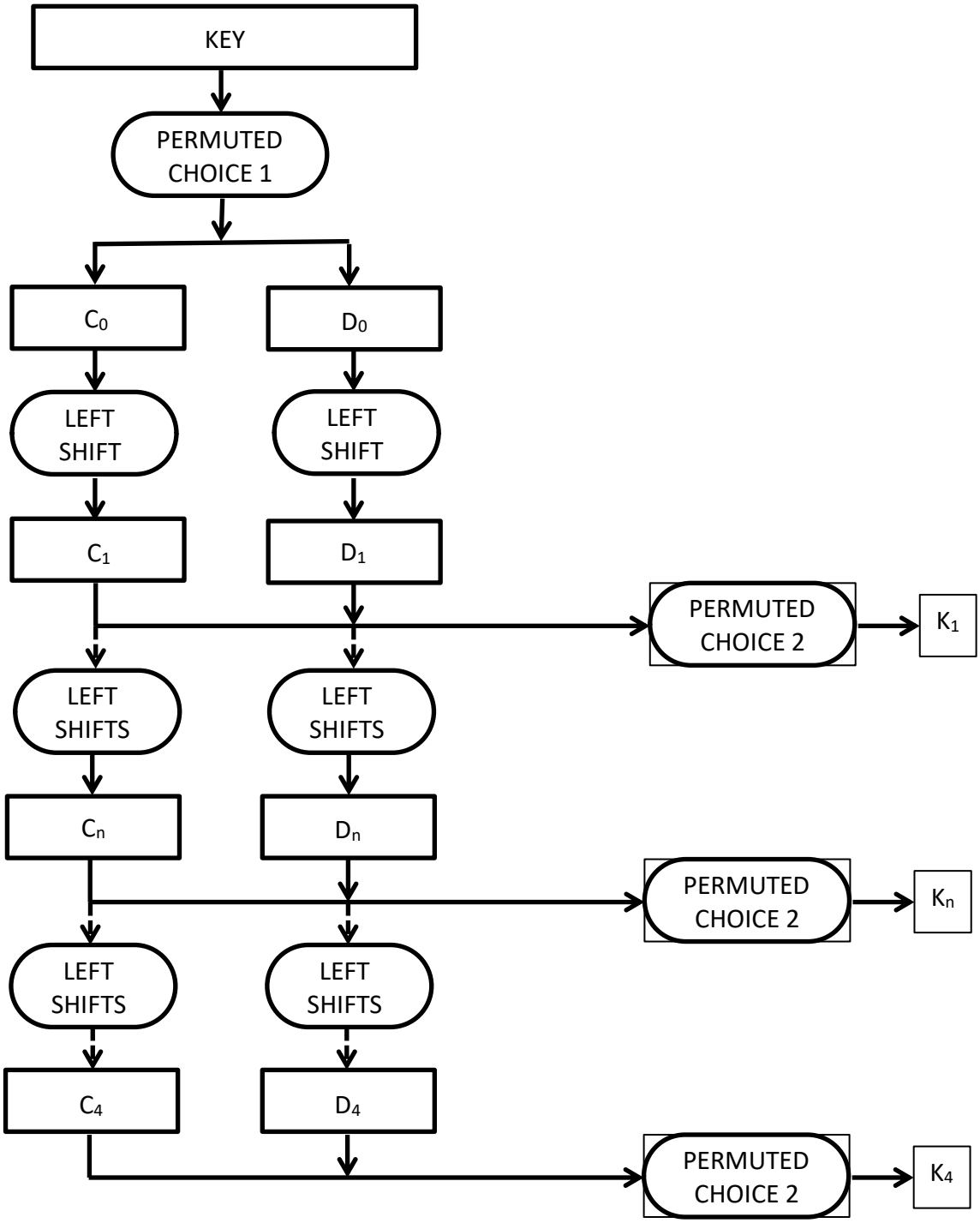


Figure 3. ***Key schedule calculation***

Iteration Number	Number of Left Shifts
1	1
2	2
3	2
4	2

For example, C_2 and D_2 are obtained from C_1 and D_1, respectively, by two left shifts, and C_1 and D_1 are obtained from C_0 and D_0, respectively, by one left shift. In all cases, by a single left shift is meant a rotation of the bits one place to the left, so that after one left shift the bits in the 5 positions are the bits that were previously in positions 2, 3,..., 5, 1.

Permuted choice 2 is determined by the following table:

PC-2

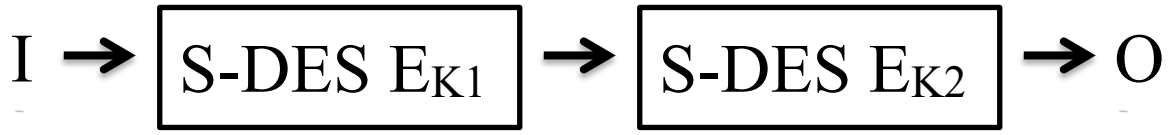
6 3 7 4 8 5 10 9

Therefore the first bit of K_n is the 6th bit of C_nD_n, the second bit is the 3rd, and so on with the 7th bit the 10th, and the 8th bit the 9th.

APPENDIX 2

DOUBLE S-DES BLOCK DIAGRAM
(ECB Mode)

DS-DEA Encryption Operation:



DS-DEA Decryption Operation: