Advances in Networks
2015; 3(3-1): 7-22
Published online September 16, 2015 (http://www.sciencepublishinggroup.com/j/net)
doi: 10.11648/j.net.s.2015030301.12
ISSN: 2326-9766 (Print); ISSN: 2326-9782 (Online)

Implementation of Software-Efficient DES Algorithm

Mohammad Taghipour[1, *], Arash Moghadami[2], Behbood Moghadam Naghdi Shekardasht[2]

[1]Industrial Engineering, Science & Research Branch of Islamic Azad University, Tehran, Iran
[2]Electrical Engineering, Telecommunications, Non-Profit Institute of Higher Education, Aba - Abyek Qazvin, Iran

Email address:
Mohamad.taghipour@srbiau.ac.ir (M. Taghipour), moghadami.arash@yahoo.com (A. Moghadami), behboodmoghadam@gmail.com (B. M. N. Shekardasht)

To cite this article:
Mohammad Taghipour, Arash Moghadami, Behbood Moghadam Naghdi Shekardasht. Implementation of Software-Efficient DES Algorithm. *Advances in Networks*. Special Issue: Secure Networks and Communications. Vol. 3, No. 3-1, 2015, pp. 7-22.
doi: 10.11648/j.net.s.2015030301.12

Abstract: By increasing development of digital telecommunication and the increase of sending and receiving data of various network of data transfer, protection of the safety of data are the most important necessities of the current world. The increase of different bank trading, increasing use of smart cards, moving to electronic government, are the examples of significance of this issue. DES algorithm is introduced by IBM company and is applied for many years by technology and standard institute of US as data encryption standard algorithm and it is also applied in many applications as networks as ATM and smart cards. Software-efficient implementation of this algorithm is one of the important research issues for engineers working in this field as we can perform rapidly on smart cards. In this study, an efficient algorithm implementation by MATLAB and C language is presented and is compared with the latest works in this field.

Keywords: Digital Telecommunication, Cryptography, Decryption, Safe Communication, DES Algorithm

1. Introduction

The development of internet has made great changes in the life style and job activity of people, organizations and institutes. The data security is one of the common issues of legal and real entities. Assurance of the lack of access of *unauthorized* users to sensitive data are the most important challenges regarding data distribution in internet. Sensitive information as hidden from other people includes various items and some of them are:

- Credit card information
- Membership numbers in associations
- Private information
- Details of personal information
- Sensitive information in an organization
- Information of bank accounts

There are not complexities of human relations as mutual trust in electronic relations and a science should prepare the conditions and these relations should be guaranteed. Thus, cryptography is the knowledge of guaranteeing the relations as without cryptography, there is no guarantee in digital communication world. Cryptography is the knowledge dealing with the principles of transfer or storage of information as secure, even if the transfer path and communication channels or data storage is insecure. Cryptography has been used to protect information, consistency of sent and received data, authentication of identity and these principles should be observed in each type of cryptography. Protecting information and confidentiality means that only the sender and receiver understands the content of message and it is possible other people can see its content but its content should be ambiguous from their view.

In the mid 70s, DES algorithm as FIPS – PUB – 46 is applied as data encryption standard and various institutes as NIST, IEEE, etc. consider it as an algorithm with suitable security for the applications without classification. Since then, hardware and software algorithm implementation is investigated on various grounds as microcontrollers, smart cards, FPGAs, etc. in various papers. For example, we can refer to the followings:

FIPS – PUB – 46-1
FIPS – PUB – 46-2
FIPS – PUB – 46-3
FIPS – PUB – 197

The purpose of this study besides recognition of function of data encryption algorithm and its simulation in MATLAB is presenting an efficient implementation of algorithm and improvement of implementation performance in comparison

with others works.

Study hypotheses are as followings:

- The structure of algorithm and its key length are defined.
- The implementation is PC and implementation language is MATLAB and C.
- Having common calculation ability
- Initial recognition of architectures and methods of algorithm implementation of reported previous works
- Theoretical basics

2. Cryptography

With the advent of computer and increasing their calculation ability, cryptography entered computer sciences and this caused three major changes in cryptography:

- High calculation power allowed the possibility that complex and effective methods are created for cryptography.
- The cryptography methods that were used for Ciphering the message had various and new applications.
- Before that, cryptography was mostly on text data and alphabets but after computer, cryptography was performed on different data and based on bit.

2.1. The History and Structure of DES Algorithm

DES algorithm was raised in 70s in US as a coding standard. This algorithm accepts a series of main text with constant length as input and after doing complex works on its, output is produced with the length equal to input length. A key is used for encryption and only those who know the key value can decode it. Although some analyses are made about DES more than any other block encryption method, the most practical attack against this algorithm is comprehensive search of key space. There are three theoretical attacks for this algorithm as needing less time compared to comprehensive searching of key space but these methods are not possible.

2.2. DES Encryption

Data encryption standard (DES) is a math algorithm applied for cryptography and Decryption of coded binary data. Cryptography converts data to Cipher. Decryption of cipher, returns to the main data. The mentioned algorithm defines both cryptography and Decryption based on binary value as key. The data are recovered by cipher, if the same key is used for Decryption as it was used for encryption. DES algorithm is composed of two:

DES cryptography algorithm

DES algorithm includes some repetitions of a plain transformation by substitution and transposition techniques. This algorithm only applies one key for encryption and decryption and it is called private key encryption. Thus, keeping the key as confidential by sender and transmitter is of great importance as the algorithm is public and in case of revealing key, any person can see the confidential message. Thus, in DES cryptography, the life of key is as long as the life of transaction.

DES cryptography key

DES key is an 8-bit sequence and each bit includes a7-bit key and a parity bit. During cryptography, DES algorithm breaks the main text into 64-bit blocks. This algorithm works on a block and it is broken into half and encryption is done character to character. Characters are permuted 16 times under the key supervision and finally an encrypted 64-bit text is produced. The key with 56 bit is useful and 8-bit is for parity.

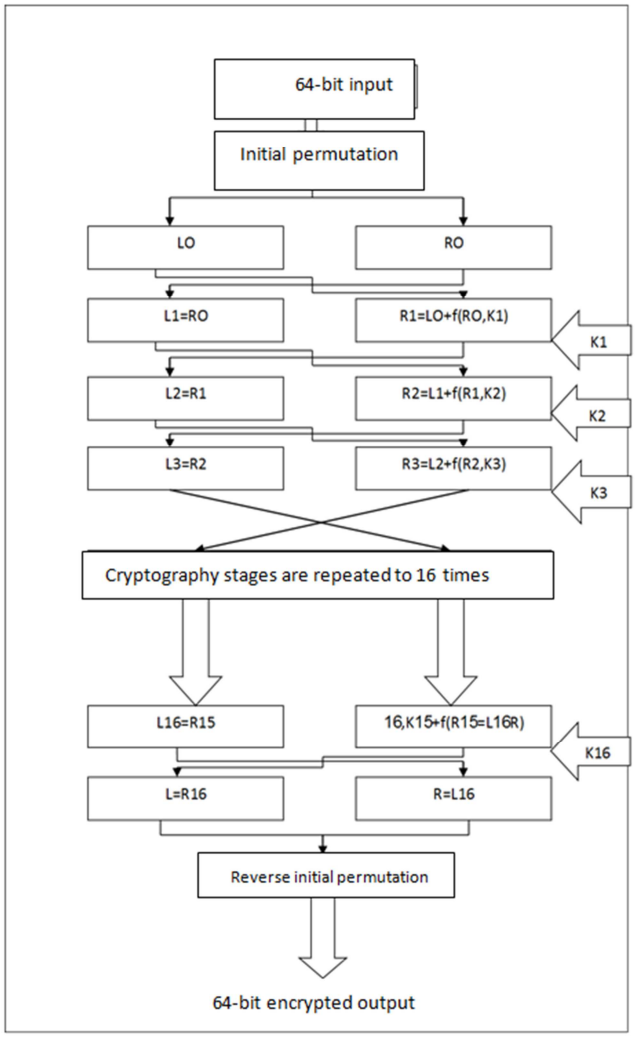

Figure 2-1. *Algorithm of DES cryptography method DES algorithm structure.*

In DES, the length of blocks is 64 bit. The key consists of 64 bit but only 56 bits are used and other 8 bits are only used to check parity. The algorithm includes 16 simialr rounds and each stage is called a round. The text being encrypted is at first subject to initial permutation (IP). Then, a series of complex acts on done on the key and finally it is subject to final permutation (FP). IP and FP are inverses. FP undoes the action of IP. IP and FP have no cryptographic significance, but were included in order to facilitate loading blocks in mid-1970s 8-bit based hardware but DES was performed slow in

software [5]. Before the main round, the block is divided into two 32-bit halves and processed alternately; this criss-crossing is known as the Feistel scheme. The Feistel structure ensures that decryption and encryption are very similar processes, the only difference is that the subkeys are applied in the reverse order when decrypting. The rest of the algorithm is identical. This greatly facilitates implementation, particularly in hardware, as there is no need for different encryption and decryption algorithms.

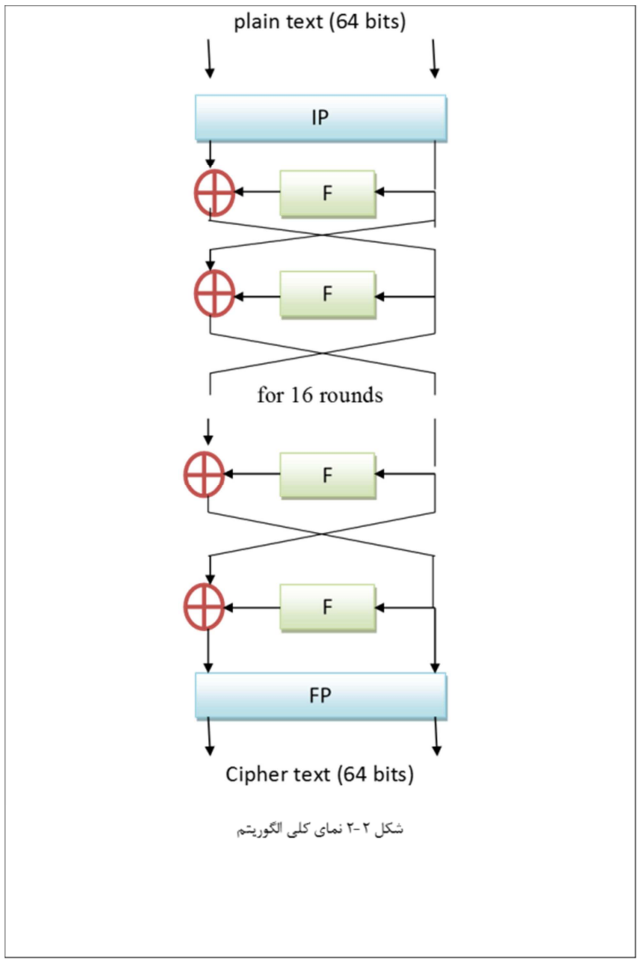

Figure 2-2. *General view of algorithm.*

The function including IP output and providing FP after 16 input rounds is called function F. This function has a 32-bit input and a 48-bit input and a 32-bit output. The entrance block is consists of 32 bits and left half and is denoted by L and other 32 bits as right half are denoted by R and the entire block is denoted by LR.

If K is a 48-bit block derived of main 64-bit key and output of a round with input LR and output L1R1 are defined as L1=R R1=L XOR F(R, K). If KS is a function with 64-bit key and an integer ranging 1-16 as input, 48-bit key of KN generates an output as KN bits are obtained by KEY bits and we have KN= KS (N.KEY):

KS is called Key schedule function. Thus, we have:

LN=Rn-1 RN =Ln-1 XOR f (Rn-1, KN)

For decryption we have: R=L1 L=R1 XOR f(L1,K). Thus, decryption is done with the same algorithm as applied for decryption and in each stage, the same k-bit is used as key for decryption and we can say:

RN-1=LN LN-1=RN XOR f (LN, KN)

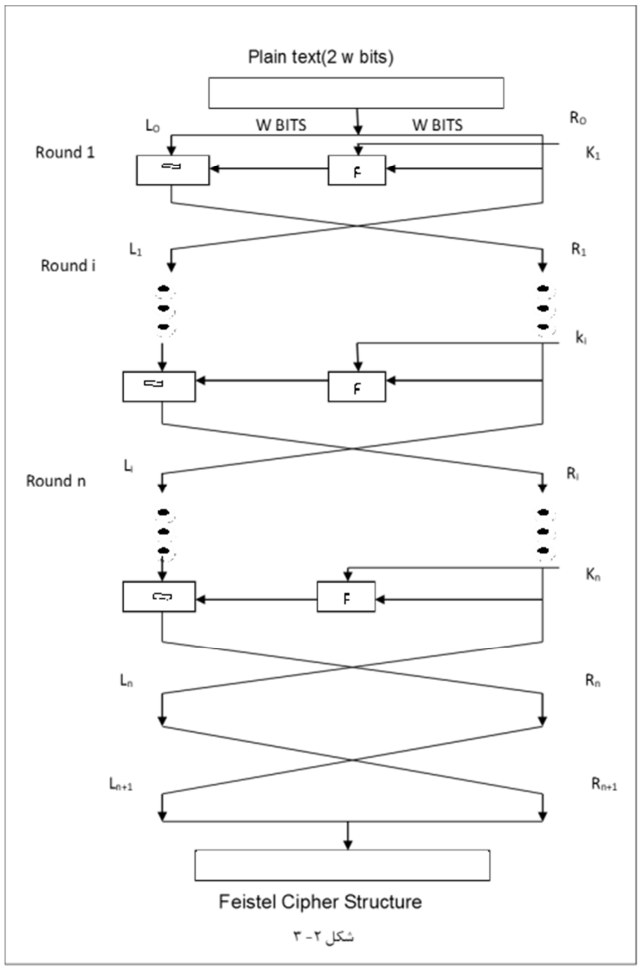

Figure 3-2. *Feistel Cipher Structure*

To compute decryption R16L16 is input of and R0L0 is input of FP. The 16th key in the first round, 15th key in the second round and the first key in 16th round can be used [3].

3. Simulation and Implementation of DES Algorithm

3.1. Step by Step DES Code

In this algorithm a 64-bit block is received from data (plaintext) and then a coded 64-bit block is delivered (cipher text).

This algorithm is composed of 16 rounds, the main algorithm is coded to generate data and is repeated 16 times.

An important point regarding the above key: in DES algorithm, the coefficients of 8th bit of key are not used, it means that bits 8, 16, 24, 32, 40, 48, 56, 64 are not considered (thus, effective length is 56-bit key).

Now, DES algorithm starts:

First stage

Creating 16 subkeys, each with the length of 48bits

At first, it is required to be familiar with Permuted choice 1 as PC-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

Since the first entry in the table is 57, this means that the 57th bit of the original key K becomes the first bit of the permuted key (K+). The 49th bit of the original key becomes the second bit of the permuted key and this continues. Thus, based on the above values, we have:

K = 00010011 00110100 01010111 01111001 10011011 10111100 11011111

11110001

K---> (PC-1)--->K+

K+ = 1111000 0110011 0010101 0101111 0101010 1011001 1001111 0001111

Second stage

Then K+ is divided into two 28-bit halves:

C0 = 1111000 0110011 0010101 0101111

D0 = 0101010 1011001 1001111 0001111

As shown, in this stage, 16 sub-keys are created (Cn,Dn) (n is ranging 1-16). These 16 subkeys are generated of D0, C0 as followings.

At first, consider the following Table:

Iteration Number	Number of Rotate left shift
1	1
2	1
3	2
4	2
5	2
6	2
7	2
8	2
9	1
10	2
11	2
12	2
13	2
14	2
15	2
16	1

Recalling:

1111000011001100101010101111 (1 Rotate left shift) 1110000110011001010101011111

Based on C0, D0 and above shifts Table, we have:

C0 = 1111000011001100101010101111

D0 = 0101010101100110011110001111

C1 = 1110000110011001010101011111

D1 = 1010101011001100111100011110

C2 = 1100001100110010101010111111

D2 = 0101010110011001111000111101

C3 = 0000110011001010101011111111

D3 = 0101011001100111100011110101

C4 = 0011001100101010101111111100

D4 = 0101100110011110001111010101

C5 = 1100110010101010111111110000

D5 = 0110011001111000111101010101

C6 = 0011001010101011111111000011

D6 = 1001100111100011110101010101

C7 = 1100101010101111111100001100

D7 = 0110011110001111010101010110

C8 = 0010101010111111110000110011

D8 = 1001111000111101010101011001

C9 = 0101010101111111100001100110

D9 = 0011110001111010101010110011

C10 = 0101010111111110000110011001

D10 = 1111000111101010101011001100

C11 = 0101011111111000011001100101

D11 = 1100011110101010101100110011

C12 = 0101111111100001100110010101

D12 = 0001111010101010110011001111

C13 = 0111111110000110011001010101

D13 = 0111101010101011001100111100

C14 = 1111111000011001100101010101

D14 = 1110101010101100110011110001

C15 = 1111100001100110010101010111

D15 = 1010101010110011001111000111

C16 = 1111000011001100101010101111

D16 = 0101010101100110011110001111

Now, we apply PC-2 on above pairs to create Kn. This permutation Table 2 is as followings:

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

Based on the above calculations we have:

C1D1 = 1110000 1100110 0101010 1011111 1010101 0110011 0011110 0011110

Now, PC-2 should be applied independently on one by one of 16 sub-keys, for example in K1 (as shown in the above Table), the first bit is equal to the 14^{th} C1D1 bit, the second bit K1 equal to the 17^{th} C1D1 bit and so on.

Finally, after applying PC-2 on calculated CnDn, we have:

K1 = 000110 110000 001011 101111 111111 000111 000001 110010

K2 = 011110 011010 111011 011001 110110 111100 100111 100101

K3 = 010101 011111 110010 001010 010000 101100 111110 011001

K4 = 011100 101010 110111 010110 110110 110011 010100 011101

K5 = 011111 001110 110000 000111 111010 110101 001110 101000

K6 = 011000 111010 010100 111110 010100 000111

101100 101111

K7 = 111011 001000 010010 110111 111101 100001 100010 111100

K8 = 111101 111000 101000 111010 110000 010011 101111 111011

K9 = 111000 001101 101111 101011 111011 011110 011110 000001

K10 = 101100 011111 001101 000111 101110 100100 011001 001111

K11 = 001000 010101 111111 010011 110111 101101 001110 000110

K12 = 011101 010111 000111 110101 100101 000110 011111 101001

K13 = 100101 111100 010111 010001 111110 101011 101001 000001

K14 = 010111 110100 001110 110111 111100 101110 011100 111010

K15 = 101111 111001 000110 001101 001111 010011 111100 001010

K16 = 110010 110011 110110 001011 000011 100001 011111 110101

Third stage

If we summarize Subkey, we have

C [0]D[0] = PC1(key)

for 1 <= i <= 16

C[i] = LS[i](C[i-1])

D[i] = LS[i](D[i-1])

K[i] = PC2(C[i]D[i])

Then, we start the coding of 64-bit blocks (DES Core Function). Consider the following Table:

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 table here means initial permutation and is applied on initial message, M.

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

IP = 1100 1100 0000 0000 1100 1100 1111 1111 1111 0000 1010 1010 1111 0000 1010 1010

Like the apply methods of previous tables, 58th bit of M here as "1" is the first IP bit, then 50th bit of M here is "1" and it is the second IP bit and so on. Then, IP is divided into two 32-bit left and right halves:

L0 = 1100 1100 0000 0000 1100 1100 1111 1111

R0 = 1111 0000 1010 1010 1111 0000 1010 1010

The general formula of this 16 rounds is as followings:

L[n] = R [n-1]

R[n] = L[n-1] XOR f(R[n-1],K[n])

Here, n is ranging 1, 16 rounds. Regarding f function, we explain later.

For example, for N=1, we have:

The items calculated are as:

L [0] = 1100 1100 0000 0000 1100 1100 1111 1111

R [0] = 1111 0000 1010 1010 1111 0000 1010 1010

K [1] = 000110 110000 001011 101111 111111 000111 000001 110010

Then,

L [1] = R[0] = 1111 0000 1010 1010 1111 0000 1010 1010

R [1] = L[0] + f(R[0],K[1])

The rest is regarding explanation of f function performance in above equation:

To compute f, at first 32-bit R[n-1] should be extended to 48bit. To do this, selection table as E is used: E(R[n-1])

The above function has input bit with 48-bit output. The table is as:

E BIT-SELECTION 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

Calculation of function f continues…

Fourth stage

In the continuance of previous section, we calculate E(R [0]) from R[0]:

R [0] = 1111 0000 1010 1010 1111 0000 1010 1010

E(R [0]) = 011110 100001 010101 010101 011110 100001 010101 010101

As shown, by apply of Table E, each four initial bits are extended to 6 bits.

In the continuance of calculation of output f E(R[n-1]) with key K[n] is XOR. for example,

K [1] = 000110 110000 001011 101111 111111 000111 000001 110010

E(R [0]) = 011110 100001 010101 010101 011110 100001 010101 010101

K1 XOR E(R [0]) = 011000 010001 011110 111010 100001 100110 010100 100111

Calculation of f is not completed yet. Here, E(R[n]) is extended from 32 bit to 48 bit by Table E. Then the result with K[n] is XOR. Now 48bit or eight 6-bit group is the result. We attribute some names as B [1] to B[8] to these 8 groups. It means that:

K[n] XOR E(R[n]-1) =B [1] B [2] B[3] B[4] B[5] B[6] B[7] B[8]

Later we refer to another operation on B[i]. Here another concept as SBoxes is presented. These Tables should be applied on B[i]. It means that:

S1(B[1]) S2(B[2]) S3(B[3]) S4(B[4]) S5(B[5]) S6(B[6]) S7(B[7]) S8(B[8])

The Tables are as:

S1
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

S2

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
S3
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
S4
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
S5
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
S6
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
S7
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
S8
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 applying method of S Boxes as different from other Tables as evaluated in the next section.

Fifth stage

The application method of S-boxes or Substitution boxes

Let we have 48-bit binary value and we want to apply S-boxes on it:

011101000101110101000111101000011100101101011101

As it was said, eight 6-bit groups are extracted from it composed of B1 to B8 as:

011101 000101 110101 000111 101000 011100 101101 011101

The following calculations are considered:

B[n] => S[n][row][column]

B[1] => S[1](01, 1110) = S[1][1][14] = 3 = 0011

B[2] => S[2](01, 0010) = S[2][1][2] = 4 = 0100

B[3] => S[3](11, 1010) = S[3][3][10] = 14 = 1110

B[4] => S[4](01, 0011) = S[4][1][3] = 5 = 0101

B [5] => S [5](10, 0100) = S[5][2][4] = 10 = 1010

B [6] => S [6](00, 1110) = S[6][0][14] = 5 = 0101

B [7] => S [7](11, 0110) = S[7][3][6] = 10 = 1010

B [8] => S [8](01, 1110) = S[8][1][14] = 9 = 1001

Calculation method:

B[n] => S[n] [row][column]

n: Is equal to B index

Row: It is created of putting the first and final bit of a 6-bit group.

Column: The rest of bits creates column (from bits 2-5)

For example:

Consider 011101

As the first 6-bit group is our 48-bit value, n=1.

To create row, two first and final bits are together, ROW=01

The column is composed of bit 2 to 5, Column=1110

Result:

The first calculation as mentioned in the above:

S[n][row][column] = S[1](01,1110)

To compute this situation in Table S1, at first the values in parenthesis are converted into their decimal equivalent and we have:

S[1][1][14]

S1 Table is as followings (For easy reference, the Number of rows and columns is written, 0-3 as number of rows and 0-15 as the number of columns (Blue) :

S1 (ROW/Column)

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
0 14 4 13 1 2 15 11 8 3 10 6 12 5 9 0 7
1 0 15 7 4 14 2 13 1 10 6 12 11 9 5 3 8
2 4 1 14 8 13 6 2 11 15 12 9 7 3 10 5 0
3 15 12 8 2 4 9 1 7 5 11 3 14 10 0 6 13

Then, this situation is found in Table S1. We refer to S1 Table and we find the value in row 1 and column. This value is equal to 3 and then it is converted to binary value. Thus, we have:

B[1] => S[1](01, 1110) = S[1][1][14] = 3 = 0011

Then, it is applied for other 6-bit groups.

The result is putting together the above results as a 32-bit binary value.

Sixth stage

After applying substitution Tables or S-boxes on Bis, another permutation is performed on the result.

Permutation 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

Briefly, f function is computed as:

f = P(S [1](B[1])...S[8](B[8]))

For example:

S1(B1)S2(B2)S3(B3)S4(B4)S5(B5)S6(B6)S7(B7)S8(B8) = 0101 1100 1000 0010 1011

0101 1001 0111

==>

f = 0010 0011 0100 1010 1010 1001 1011 1011

Then,

R [1] = L[0] XOR f(R[0], K[1])

= 1100 1100 0000 0000 1100 1100 1111 1111

XOR 0010 0011 0100 1010 1010 1001 1011 1011

= 1110 1111 0100 1010 0110 0101 0100 0100

L1 was computed already:

L [1] = R[0] = 1111 0000 1010 1010 1111 0000 1010 1010

Here, one round of 16 rounds of algorithm is finished.

Start of second round:

According to the general formula we have:

L [2] = R [1]

R [2] = L [1] + f(R [1], K [2])

Its computation is simple and this trend is continued to the end of 16 rounds.

At the end of round 16, the result is L16, R16. Here, these two blocks are changed, it means that

R [16] L [16]

And the final permutation is performed on it as following Table:

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

For example, based on the selected values we have:

L16 = 0100 0011 0100 0010 0011 0010 0011 0100

R16 = 0000 1010 0100 1100 1101 1001 1001 0101

===>

R [16]L[16] = 00001010 01001100 11011001 10010101 01000011 01000010 00110010

00110100

===>

IP^-1 = 10000101 11101000 00010011 01010100 00001111 00001010 10110100

00000101 (bin)

= 85E813540F0AB405 (hex)

Or briefly:

M = 0123456789ABCDEF

The applied key for coding:

Key = 13 34 57 79 9B BC DF F1

Encoded output

C = 85E813540F0AB405

3.2. Simulated Code of DES Algorithm in MATLAB

```
Clear
tic
M= 'AF88888888555456;’
K = '1256984563214569;’
MB;[]=
for i=1;16
Mi=M(I;(
MBi=[‘0000’,,dec2bin(hex2dec(Mi;[((
MBi=MBi(end-3;end;(
MBi=[str2num(MBi(1)),str2num(MBi(2)),str2num(MBi(3)),str2num(MBi(4;](((
MB=[MB,MBi ;[
end
M=MB;
KB ;[]=
for i=1;16
Ki=K(i ;(
KBi=[‘0000',dec2bin(hex2dec(Ki ;[((
KBi=KBi(end-3; end;(
KBi=[str2num(KBi(1),str2num(KBi(2)),str2num(KBI(3)),str2num(KBi(4 ;[(((
KB=[KB,KBi;[
end
K=KB;
E=[32, 1, 2, 3, 4, 5;
   9, 8, 7, 6, 5, 4;
   9, 10, 11, 12, 13, 8;
   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;
   S1=14, 4, 13, 1, 2, 15, 11, 8, 3, 10, 6, 12, 5, 9, 0, 7;
   8, 3, 5, 9, 11, 12, 6, 10, 1, 13, 2, 14, 4, 7, 15, 0;
   0, 5, 10, 3, 7, 9, 12, 15, 11, 2, 6, 13, 8, 14, 1, 4;
   13, 6, 0, 10, 14, 3, 11, 5, 7, 1, 9, 4, 2, 8, 12, 15;
   S2= 15, 1, 8, 14, 6, 11, 3, 4, 9, 7, 2, 13, 12, 0, 5, 10;
   5, 11, 9, 6, 10, 1, 0, 12, 14, 8, 2, 15, 7, 4, 13, 3;
```

```
   15, 2, 3, 9, 6, 12, 8, 5, 1, 13, 4, 10, 117, 14, 0;
   9, 14, 5, 0, 12, 7, 6, 11, 2, 4, 15, 3, 1, 10, 8, 13;
   S3=10, 0, 9, 14, 6, 3, 15, 5, 1, 13, 12, 7, 11, 4, 2, 8;
P=[16,7,20,21,29,12,28,17,1,15,23,26,5,18,31,10,2,8,24,14,32,27,3,9,19,13,30,6,22,1 1,4,25 ;[

PC1=[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 ;[

PC2=[14,17,11,24,1,5,3,28,15,6,21,10,23,19,12,4,26,8,16,7,27,20,13,2,41,52,31,37,4
7,55,30,40,51,45,33,48,44,49,39,56,34,53,46,42,50,36,29,32 ;[

Ki=zeros(16,48 ;(

K_PC1=K(PC1                ;(

CO=K_PC1(1;28 ;(

DOK_PC1(29;56 ;(

for i=1;16

if i==1| | i==2| | i==9| |I==16


   1, 15, 11, 12, 14, 5, 8, 2, 10, 6, 4, 3, 9, 0, 7, 13;
   7, 14, 10, 5, 12, 2, 1, 11, 0, 3, 15, 8, 9, 4, 6, 13;
   12, 2, 5, 11, 3, 14, 15, 47, 8, 9, 6, 0, 13, 10, 1;
   S4= 7, 13, 14, 3, 0, 6, 9, 10, 1, 2, 8, 5, 11, 12, 4, 15;
   9, 14, 10, 1 12, 2, 7, 4, 3, 0, 15, 6, 5, 11, 8, 13;
   4, 8, 2, 5, 14, 3, 1, 15, 137, 11, 12, 0, 9, 6, 10;
   14, 2, 7, 12, 11, 5, 4, 9, 8, 13, 1, 10, 6, 0, 15, 3;
   S5= 2, 12, 4, 1, 7, 10, 11, 6, 8, 5, 3, 15, 13, 0, 14, 9;
   6, 8, 9, 3, 10, 15, 0, 5, 1, 13, 7, 4, 12, 2, 11, 14;
   14, 0, 3, 6, 5, 12, 9, 15, 8, 7, 13, 10, 11, 1, 2, 4;
   3, 5, 4, 10, 9, 0, 15, 6, 13, 2, 14, 1, 7, 12, 8, 11;
   S6=12, 1, 10, 15, 9, 2, 6, 8, 0, 13, 3, 4, 14, 7, 5, 11;
   8, 3, 11, 0, 14, 13, 1, 6, 5, 9, 12, 7, 2, 4, 15, 10;
   6, 11, 13, 1, 10, 4, 0, 7, 3, 12, 8, 2, 5, 15, 14, 9;
   13, 8, 0, 6, 7, 1, 14, 11, 10, 15, 5, 9, 12, 2, 3, 4;
   S7=4, 11, 2, 14, 15, 0, 8, 13, 3, 12, 9, 7, 5, 10, 6, 1;
   6, 8, 15, 2, 12, 5, 3, 14, 10, 1, 9, 4, 7, 11, 0, 13;
   2, 9, 5, 0, 8, 6, 15, 10, 14, 7, 3, 12, 13, 11, 4, 1;
   12, 3, 2, 14, 15, 0, 5, 9, 7, 10, 4, 1, 8, 13, 11, 6;
   S8=13, 2, 8, 4, 6, 15, 11, 1, 10, 9, 3, 14, 5, 0, 12, 7;
   2, 9, 14, 0, 11, 6, 5, 12, 4, 7, 3, 10, 8, 13, 15, 1;
   8, 5, 3, 15, 13, 10, 6, 0, 2, 14, 12, 9, 1, 4, 11, 7;
   11, 6, 5, 3, 0, 9, 12, 15, 13, 8, 10, 4, 7, 14, 1, 2;
P=[16,7,20,21,29,12,28,17,1,15,23,26,5,18,31,10,2,8,24,14,32,27,3,9,19,13,30,6,22,1 1,4,25 ;[

PC1=[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 ;[
```

PC2=[14,17,11,24,1,5,3,28,15,6,21,10,23,19,12,4,26,8,16,7,27,20,13,2,41,52,31,37,4
7,55,30,40,51,45,33,48,44,49,39,56,34,53,46,42,50,36,29,32 ;[

Ki=zeros(16,48 ;(

K_PC1=K(PC1 ;(

CO=K_PC1(1;28 ;(

DO=K_PC1(29;56 ;(

for i=1;16

if i==1| | i==2| | i==9| |I==16

1, 15, 11, 12, 14, 5, 8, 2, 10, 6, 4, 3, 9, 0, 7;
7, 14, 10, 5, 12, 2, 1, 11, 0, 3, 15, 8, 9, 4, 6, 13;
12, 2, 5, 11, 3, 14, 15, 4, 7, 8, 9, 6, 0, 13, 10, 1;
S4=7, 13, 14, 3, 0, 6, 9, 10, 1, 2, 8, 5, 11, 12, 4, 15;
9, 14, 10, 1, 12, 2, 7, 4, 3, 0, 15, 6, 5, 11, 8, 13;
4, 8, 2, 5, 14, 3, 1, 15, 13, 7, 11, 12, 0, 9, 6, 10;
14, 2, 7, 12, 11, 5, 4, 9, 8, 13, 1, 10, 6, 0, 15, 3;
S5=2, 13, 4, 1, 7, 10, 11, 6, 8, 5, 3, 15, 13, 0, 14, 9;
6, 8, 9, 3, 10, 15, 0, 5, 1, 13, 7, 4, 13, 2, 11, 14;
14, 0, 3, 6, 5, 12, 9, 15, 8, 7, 13, 10, 11, 1, 2, 4;
3, 5, 4, 10, 9, 0, 15, 6, 13, 2, 14, 1, 7, 12, 8, 11;
S6=12, 1, 10, 15, 9, 2, 6, 8, 0, 13, 3, 4, 14, 7, 5, 11;
8, 3, 11, 0, 14, 13, 1, 6, 5, 9, 12, 7, 2, 4, 15, 10;
6, 11, 13, 1, 10, 4, 0, 7, 3, 12, 8, 2, 5, 15, 14, 9;
13, 8, 0, 6, 7, 1, 14, 11, 10, 15, 5, 9, 12, 2, 3, 4;
S7=4, 11, 2, 14, 15, 0, 8, 13, 3, 12, 9, 7, 5, 10, 6, 1;
6, 8, 15, 2, 12, 5, 3, 14, 10, 1, 9, 4, 7, 11, 0, 13;
2, 9, 5, 0, 8, 6, 15, 10, 14, 7, 3, 12, 13, 11, 4, 1;
12, 3, 2, 14, 15, 0, 5, 9, 7, 10, 4, 1, 8, 13, 11, 6;
S8=13, 2, 8, 4, 6, 15, 11, 1, 10, 9, 3, 14, 5, 0, 12, 7;
2, 9, 14, 0, 11, 6, 5, 12, 4, 7, 3, 10, 8, 13, 15, 1;
8, 5, 3, 15, 13, 10, 6, 0, 2, 14, 12, 9, 1, 4, 11, 7;
11, 6, 5, 3, 0, 9, 12, 15, 13, 8, 10, 4, 7, 14, 1, 2;

```
D0=[D0(2;end),D0(1                                        ;[s
else
CO=[CO(3;end),CO(1.2 ;[(
D0=[DO(3;end),DO(1.2 ;[(
end
K_LS=[CO,D0 ;(
Ki(i,;)=K_LS(PC2 ;(
end
L=M(1;32 ;(
R=M(33;64;(
for i=1;16
E0=reshape(E',1,48 ;(
R_E=R(EO                                                  ;(
R_Ki=mod(R_E+Ki(i,;),2 ;(

B=R_Ki(1;6;(

x=B(1)*2+B(6)+1
y=B(2)*8+B(3)*4+B(4)*2+B(5)+1
C=['0000',aec2bin(S1(x,y [((
C=C(end-3;end ;(
C1=[str2num(C(1)),str2num(C(2)),str2num(C(3)), str2num(C(4 ;(([

B=R_Ki(7;12;(
x=B(1)*2+B(6)+1 ;
y=B(2)*8+B(3)*4+B(4)*2+B(5)+1 ;
C=['0000',aec2bin(S2(x,y [((
C=C(end-3;end ;(
C2=[str2num(C(1)),str2num(C(2)), str2num(C(3)), str2num(C(4 ;(([
B=R_Ki(13;18;(
x=B(1)*2+B(6)+1 ;
y=B(2)*8+B(3)*4+B(4)*2+B(5)+1 ;
C=['0000', dec2bin(S3(x,y [((
C=C(end-3;end (
```

```
C3=[str2num(C(1)),str2num(C(2)),str2num(C(3)), str2num(C(4 ;(([

B=R_Ki(19;24;(
x=B(1)*2+B(6)+1 ;
y=B(2)*8+B(3)*4+B(4)*2+B(5)+1 ;
C=['0000", dec2bin(S4(x,y [((
C=C(end-3;end ;(
C4=[str2num(C(1)),str2num(C(2)),str2num(C(3)), str2num(C(4 ;(([

B=R_Ki(25;30;(
x=B(1)*2+B(6)+1 ;
y=B(2)*8+B(3)*4+B(4)*2+B(5)+1 ;
C=['0000", dec2bin(S5(x,y [((
C=C(end-3;end ;(
C5=[str2num(C(1)), str2num(C(2)), str2num(C(3)), str2num(C(4 ;[((
B=R_Ki(31;36 (
x=B(1)*2+B(6)+1 ;
y=B(2)*8+B(3)*4+B(4)*2+B(5)+1 ;
C=['0000', dec2bin(S6(x,y [((
C=C(end-3;end ;(
C6=[str2num(C(1)),str2num(C(2)),str2num(C(3)), str2num(C(4 ;[((

B=R_Ki(37;42 ;(
x=B(1)*2+B(6)+1 ;
y=B(2)*8+B(3)*4+B(4)*2+B(5)+1 ;
C=['0000", dec2bin(S7(x,y [((
C=C(end-3;end ( C7=[str2num(C(1)),str2num(C(2)),str2num(C(3)), str2num(C(4 ;[[(

B=R_Ki(43;48;(
x=B(1)*2+B(6)+1 ;
y=B(2)*8+B(3)*4+B(4)*2+B(5)+1 ;
C=['0000',aec2bin(S8(x,y ;[((
C=C(end-3;end ( C8=[str2num(C(1)),str2num(C(2)), str2num(C(3)), str2num(C(4 ;[[(
C=[C1,C2, C3,C4,C5,C6,C7, C8 ;[
```

```
R_P=C(P ;(

TEMP=L;

L=R ; R=mod(TEMP+R_P,2 ;(

end

TEMP=L;

L=R ;

R=TEMP ;

C=[L,R ;[

CS ;[]-

C=num2str(C (

pos=find(C ;("--

C=C(pos ;(

for i=1;4;61

Ci=C(i;i-H3 (

CS=[CS, num2str(dec2hex(bin2dec(Ci ;[(((

end

C=CS

t=toc

C= B49584B65CD90953

T=0.2895
```

After performing the above code in MATLAB, we have R=TEMP

3.3. Implementation of DES Algorithm in C++ Application

```
#include;stdio.h>
#include;fstream.h>
#include;string.h>
#include;conio.h>
#include;iostream.h>
        int key(64]={
        0,0,0,1,0,0,1,1,
        0,0,1,1,0,1,0,0,
        0,1,0,1,0,1,1,1,
        0,1,1,1,1,0,0,1,
        1,0,0,1,1,0,1,1,
        1,0,1,1,1,1,0,0,
        1,1,0,1,1,1,1,1,
        1,1,1,1,0,0,0,1
        };
class Des
{
public;
int
```

```
keyi[16][48), total[64), left|32), right|32), ck[28), dk[28), expansion[48),z[48),xor1[48), sub[32],p[32],xor2[32],temp[64],
pc1[56],ip[64],inv[8][8];
char final[1000];
void IP();
void PermChoice1();
void PermChoice2();
void Expansion();
void inverse();
void xor two();
void xor oneE(int);
void xor oneD(int);
void substitution();
void permutation();
void keygen();
char* Encrypt(char *);
char * Decrypt(char *);
};
void Des;;IP()
{
        int k=58,i;
        for(i=0;i < 32;i++)
        {
                ip[i]=total[k-1];
                if(k-8>0) k=k-8;
                else k=k+58;
        }
        k=57;
        for(i-32;i < 64;i++)
        {
                ipsi]=total[k-1];
                if(k-8 > 0) k=k-8;
                else        k=k+58;
         }
}
void Des;;PermChoice1()
{
        int k=57,i;
        for(i-0;i < 28;i++)
        {
                pc1[i]=keysk-1];
                if(k-8 > 0) k=k-8;
                else       k=k+57;
        }
        k=63;
        for(i-28;i < 52;i++)
        {
                pc1(i)=keysk-1];
                if(k-8 > 0) k=k-8;
                else       k=k+55;
        }
        k=28;
        for(i=52;i < 56;iłł)
        {
                pc1(i)=keysk-1];
                k=k-8;
        }
}
void Des;;Expansion()
{
```

```
        int exp(8][6],i,j,k;
        for(i-0;i < 8;i++)
        {
                for(j=0;j< 6;j++)
                {
                        if((j!=0)||(j!=5))
                        {
                                k=4*i+j;
                                expsi][j]=right[k-1];
                                }
                                if(j==0)
                                {
        substitution();
        permutation();
        xor two();
        for(i-0;i < 32;i++) left[i]=right[i];
        for(i-0;i < 32;i++) right[i]=xor2|[i];
        }
        for(i-0;i < 32;i++) tempsi]=right[i];
        for(;i < 64;i++) tempsi]=left[i-32];
        inverse();
                        k=128; d-O;
                for(i-0;i < 8;i++)
                {
                   for(j=0;jo 8;j++)
                     {
                                d=d+inv[i][j]*k;
                                k=k/2;
                     }
                        final[mc++]=(char)d;
                        k=128; d-0;
                     }
}
final[mc]='\0';
return(final);
}
char * Des;;Decrypt(char *Text!)
{
int i, a1,j, nB,m, iB,k,K,BI8],n,t,d, round,
char *Text=new char(1000];
unsigned char ch;
strcpy(Text,Text1);
i=strlen(Text);
keygen();
int mo-O;
for(iB=0,nB=0,m=0;m « (strlen(Text)/8);m #4)
{
                for(iB=0, i=0;i < 8;i++,nB++)
                {
                ch=Text[nB],
                n=(int)ch ;
                for(K=7;n >= 1;K--)
                {
                B[K]=no}{2; n/=2;
                } for(;K->= 0;K--) BIK)=0;
                for(K=0;K <8;K++,iB++) total[iB]=B[K]; } |P();
                }
        IP();
        for(i-0;i < 32;i++) left[i]=total[i];
```

```
        for(;i < 64;i++) right[i-32]=total[i];
                for(round=1;round <= 16;round++)
                {
Expansion();
xor_ones) (round);
substitution();
permutation();
xor two();
for(i-0;i < 32;i++) left[i]=right[i];
for(i=0;i < 32;i++) right[i]=xor2|[i];
}
for(i-0;i < 32;i++) tempsi]=right[i];
for(;i < 64;i++) tempsi]=left[i-32];
inverse();
                k=128; d-0;
        for(i=0;i < 8;i++)
        {
        for(j=0;jo 8;j++)
        {
                d=d+inv[i][j]*k;
                k=k/2;
        }
                final[mc++]=(char)d;
                k=128; d-O;
}
} final[mc]='\0';
char *final|1=new char[1000];
for(i=0,j=strlen(Text);i < strlen(Text);i++, ++)
        final1(i)=finals]]; final1(i)="\O';
return(final);
}
int main()
{
Des d1,d2;
char *strznew char(1000];
char *str1=new char(1000];
//crls();
coutok"Enter a string ; ";
gets(str);
str1=d1.Encrypt(str);
cout<<"\n Plain Text; "<< strz< endl;
cout<<"\nCipher Text ; "<< str1<< endl;
cout<<"\nPlain Text;"<< d2.Decrypt(str1)<< endl;
getch();
return 0;
}
void Des;;keygen()
{
        Permchoice1();
        int i,j,k=0;
        for(i-0;i < 28;i++)
        {
                ck[i]=pc.1[i];
        }
        for(i=28;i < 56;iłł)
        {
                dk[k]=pc1[i);
                k++;
        }
```

```
        int noshift=0,round;
        for(round=1;round;-16;round++)
        {
                if(round==1| |round==2| |round==9] Iround==16)
                        noshift=1;
                else
                        noshift=2;
                while(noshift × 0)
                {
                        int t;
                        t=ck[0];
                        for(i=0;i < 28;i++)
                        ck[i]=ck[i+1];
                        ck[27]=t;
                        t=dk[0];
                        for(i-O;i < 28;i++)
                        dk{i}=dk[i+1);
                        dk[27)=t;
                        noshift--;
                }
                PermChoice2();
                for(i-0;i < 48;i++)
                        keyi[round-1][i]=z[i];
                }
}
```

```
Enter a string ; arash
Plain Text ; arash
cipher Text; ÓÓ||
```

4. Conclusion

DES algorithm is one of the most important and common cryptography algorithms as applied in security of many applications as financial transaction, data transfer, smart cards, etc. In software implementation of this algorithm, it is always considered by researchers and experts in this field. Despite the significance of this issue, it is not considered mostly in our country. In this thesis, besides recognition of the function of this algorithm and simulation of algorithm performance in MATLAB, an efficient implementation by C language is performed. One of the results of this thesis is providing security of various systems as sending and receiving important data without classification. Also, smart cards or various security transactions can be used. The results also can be used to implement secure and recognize algorithm of 3-DES.

References

[1] B. Schneir, Applied cryptography, John Wiley, 1996.

[2] A. Saloma, Public key cryptography, 1990.

[3] G. Brassard,"Modern cryptography, a tutorial", 1988.

[4] C. G. Shannon,"Communication theory in security systems", Bell Sys. Tech. Journal, Vol.28, Oct. 1949.

[5] W. Diffie and M. Hellman,"New directions in cryptography", IEEE trans. On IT, Vol. 22, Nov. 1976.

[6] W. Diffie and M. Hellman,"Privacy and Authentication: An introduction to cryptography, New directions in cryptography", IEEE trans, Vol. 67, 1979.

[7] J.L. Massey, "An Introduction to contemporary cryptography", IEEE Proc, Vol.76, May 1983.

[8] Chan, M.H.L and Donaldson, R.W. "Amplitude, width and interarrival distributions for noise impulses on intrabuilding power line communication networks", IEEE Trans. Electromagn. Compat. Vol. EMC-31, pp320-323, Aug 1989.

[9] Canete, F. Cort´es, J. D´iez, L and Entrambasaguas, J. "A channel model proposal for indoor power line communications," in IEEE Communications Magazine, January 2011.

[10] Marihart, D. J. "Communications Technology Guidelines for EMS/SCADA Systems" IEEE TRANS. ON POW.DELIVERY, VOL. 16, NO. 2, APRIL 2001.

[11] Dostert, K "Telecommunications over the Power Distribution Grid- Possibilities and Limitations", Proc 1997 Internat.Symp.on Power Line Comms and its Applications pp1-9, 1997.