International Journal on Emerging Technologies 11(5): 705-712(2020)
ISSN No. (Print): 0975-8364
ISSN No. (Online): 2249-3255

Data Encryption Standard Algorithm in Symmetric Key Cryptography over Finite Field F_2

Arun Kumar Sharma[1] and Nikhlesh Kumar Badoga[2]
[1]Department of Computer Science & Engineering, NIT Hamirpur, India.
[2]Department of Computer Science & Engineering, Thapar Institute of Engineering and Technology, India.

(Corresponding author: Arun Kumar Sharma)
(Received 22 October 2020, Revised 02 December 2020, Accepted 29 December 2020)
(Published by Research Trend, Website: www.researchtrend.net)

ABSTRACT: Cryptography plays a vital role in information technology and communication process. Cryptography ensures security of information. Security that ranged from ATM cards, digital passwords and electronic commerce, all rely heavily on cryptography for security. The capability and efficiency of mathematical algorithm determines the level of security that is provided to data. The main and fundamental objective of cryptography is to enable secure communication over an insecure channel. We focus briefly on cryptography and on Data Encryption Standard (DES) Algorithm in detail.

Keywords: Data Encryption Standard, Symmetric Key Cryptography, Encryption, Decryption, Confidentiality, Authenticity, Integrity, Finite Field.

I. INTRODUCTION

The encryption algorithm that is mostly used in the word is Data Encryption Standard Algorithm [4]. The two words that have been synonymous for most of the time are "secret code making" and DES. A solicitation for crypto systems was published by NIST(National institute of Standards and Technology)in federal register on May 15, 1973. This ultimately led to the adoption of the Data Encryption Standard [3]. DES was the modification of earlier system Lucifer and was developed at IBM. The publication of DES was first done in the Federal Register on March 17, 1975. After a good amount of public discussion, the adoption of DES as a standard of “unclassified” applications was carried on January 15, 1977. The durability of DES as a standard was doubted initially and was sought to be applicable only for 10-15 years but it proved to be much more durable. It was initially expected that DES would only be used as a standard for 10-15 years; however, it proved to be much more durable. Every 5 years approximately it was reviewed to check any intricacies corresponding to its adoption as a standard. Appropriate recommendation based on proper filtering of the content is very essential, see [11, 13]. DES works on binary numbers that corresponds to 0 or 1, which the digital computers are common to. The culmination of four bits makes up base 16 number or hexadecimal number. For e.g. Hexadecimal number "A" is equal to Binary“1010”.

The 64-bitmessage groups are encrypted by DES, that is the same as 16 hexadecimal numbers. If a certain case arises where the message size exceeds or falls short of 64 bits and is not even the multiple of 64 then. Certain padding schemes are used which mainly comprises of adding certain bits to fit it into the required criteria. Generally, the message is made of addition of 0, so that the message falls into the required criteria condition. For encryption, DES uses keys, which are 64 bits long or 16 hexadecimal numbers. The DES algorithm ignores every 8^{th} key bit, which makes the effective key size to be of 56-bits [9].

II. DATA ENCRYPTION STANDARD ALGORITHM

DES is a block cipher and it converts the plaintext blocks of a given size (64-bits) and returns the same size cipher text blocks [1, 5-8]. Thus the maximum possible permutations of 64 bits results in 2^{64}, each of which can be either 0 or 1. Each block comprises of a left half block Land a right half block R each of 32 bit which in total makes the 64 bits.

The key size of each 64 bits block is of 56 bits as each 8^{th} bit in the key is ignored but they are actually stored as 64 bits block with every 8^{th} bit being inactive. The keys are actually stored as being 64-bits long, but every 8^{th} bit in the key is not used. Data Encryption Standard Algorithm, [12, 14] follows various steps as described below:

Step-1. Keys Generation:

The permutation of the 64-bits key is done as described in Table 1, ($\boldsymbol{PC1}$). Since the value of the second entry is "49", which further means that$\boldsymbol{K}^*$ which is the permuted key contains the 2^{nd} bit same as is the 49^{th} bit of the original key. The 9^{th} bit of theoriginal key becomes the seventh bit of the permuted key. The 12^{th} bit of the original key is the second last bit of thepermuted key, which further leads to involving of only 56 bits in the original key. Total 16 sub keys are created in this pattern.

Table 1: $PC1\,j$.

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

Now, the key is split in two left and right halves, C_0 and D_0, with each half containing 28 bits. With C_0 and D_0 defined, we now create sixteen blocks C_n and D_n, where 1≤*n*≤16. C_{n-1} and D_{n-1} leads to the formation of new pair of blocks C_n and D_n, respectively, corresponding to values which ranges from *n*= 1, 2, ..., 16, by using the schedule that pertains to "left shifts" of previous block. Left shift is done, by moving each bit to the left by one place, except first bit, which is cycled to the end of the block.

Table 2.

Iteration Number	1	2	3	4	5	6	7	8
Number of Left Shift	1	1	2	2	2	2	2	2
Iteration Number	9	10	11	12	13	14	15	16
Number of Left Shift	1	2	2	2	2	2	2	1

This further means, for example, C_4 and D_4 further leads to formation of C_5 and D_5 respectively, by two left shifts, and C_{15} and D_{15} leading to the formation of C_{16} and D_{16}, respectively, by one left shift. The rotation of the bits by one place to the left signifies a single left shift, so that after one left shift the bits in the 28 positions are the bits that were previously in positions 2, 3,..., 28, 1.
We now form the keys K_n, for 1≤*n*≤16, by the application of the following permutation Table 3 to each of the concatenated pairs C_nD_n. Each pair has 56 bits, but $\boldsymbol{PC2}$ only uses 48 of these.

Table 3: $\boldsymbol{PC2}$.

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	25	31	37	47	55
30	40	51	45	33	48
44	49	39	56	34	53
46	42	50	36	29	32

Therefore, the first bit of K_n is the 14th bit of C_nD_n, the second bit the 17th, and so on, ending with the 48th bit of K_n being the 32th bit of C_nD_n.

Step-2. Encryption Procedure for the Original Message:

(I) Initial Permutation: The message data ***M*** which is of 64 bits has an initial permutation ***IP.*** The rearrangement of the bits take place keeping in consideration Table 4, where the new bits arrangement with respect to their initial order is shown in Table – 4. The 58^{th} bit of ***M*** now corresponds to the first ***IP*** bit. The 50th bit of ***M*** becomes the second bit of ***IP***. The 7th bit of ***M*** is the last bit of ***IP***.

Table 4: Initial Permutation.

58	50	42	34	26	18	10	02
60	52	44	36	28	20	12	04
62	54	46	38	30	22	14	06
64	56	48	40	32	24	16	08
57	49	41	33	25	17	09	01
59	51	43	35	27	19	11	03
61	53	45	37	29	21	13	05
63	55	47	39	31	23	15	07

(II) Next, the permuted block ***IP*** is divided in to a left half $\boldsymbol{L_0}$ which is of 32 bits, and a right half $\boldsymbol{R_0}$ that comprises of 32 bits. We carry similarly through 16 iterations, for 1≤*n*≤16, making use of function ***f*** that operates on two blocks in addition to data block of size 32 bits and a $\boldsymbol{K_n}$ key of 48 bits which makes a block of 32 bits. Let **+** denote **XOR** addition, (bit-by-bit addition modulo 2). Then for ***n*** going from 1 to 16, we evaluate

$$\boldsymbol{L_n = R_{n-1}}$$

and

$$\boldsymbol{R_n = L_{n-1} + f(R_{n-1}, K_n)}.$$

Thus the final block of $L_{16}R_{16}$ is obtained for ***n*** = 16.That is, for every iteration, the 32 right bits of the previous result is taken and are made the 32 left bits of the current step. For the 32 bits which are to the right taken in the present step, we XOR the 32 left bits of the previous step by calculating ***f***. Each block of $\boldsymbol{R_{n-1}}$ is expanded from 32-bits to 48-bits in order to calculate ***f***. Selection table is used that repeats some of the bits in $\boldsymbol{R_{n-1}}$. The function **E** is obtained by using the selection Table. Thus **E**($\boldsymbol{R_{n-1}}$) contains a block for input which is of 32-bit, and an output block which is of 48 bit. Let **E** be such that the output which is of 48bits, is composed of 8 blocks each of which contains 6 bits, and the output is obtained by making use of Table 5 by selecting the inputs in order. The bits which are in positions 32, 1 and 2 of $\boldsymbol{R_{n-1}}$, make the first 3 bits of **E**($\boldsymbol{R_{n-1}}$) while the last 2-bits which are in positions 32 and 1 respectively makes the last 2 bits of **E**($\boldsymbol{R_{n-1}}$). Next, for calculating **f**, the output of **E**($\boldsymbol{R_{n-1}}$) is XOR-ed with the key $\boldsymbol{K_n}$:$\boldsymbol{K_n}$+ **E**($\boldsymbol{R_{n-1}}$).

Table 5: E-bit Selection Table.

32	01	02	03	04	05
04	05	06	07	08	09
08	09	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	01

The calculation of the function ***f*** is not finished yet. To this point, the expansion of $\boldsymbol{R_{n-1}}$ is done ranging from 32 bits to 48 bits, making use of selection table, and further the result is XORed with the key $\boldsymbol{K_n}$. Now we have 48-bits, which further can be decoded as with six bits in every one of the 8 groups.

(III) Each group comprising six bits is used to describe the addresses in tables known as "**S boxes**", see also

[2,10, 15, 16]. **S** box is used to signify the address composed of each group of 6 bits. The address will be used to denote the 4-bits-number. The original 6-bits are replaced by 4-bits number. The net result is that the eight groups of 6-bits are transformed into eight groups of 4 bits (the 4-bits outputs from the **S** boxes) takes place from the 8 groups of 6 bits which makes the total of 32-bits. The 48-bits previous result is written, in the form: K_n+ E$(R_{n-1}) = B_1B_2B_3B_4B_5B_6B_7B_8$,

where each $\boldsymbol{B_i}$is a group of 48-bits. Further we make calculation of

$S_1(B_1)\,S_2(B_2)\,S_3(B_3)\,S_4(B_4)\,S_5(B_5)\,S_6(B_6)\,S_7(B_7)\,S_8(B_8)$,

where ***i***-th **S** box output is referred to as $\boldsymbol{S_i(B_i)}$.

Each functions $\boldsymbol{S_1,\ S_2,\ldots,\ S_8}$, takes a input which makes a 6 bit block and yields an output of a block comprising of 4 bits. This description is shown in the following Table 6.

Table 6: S_1 Box.

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

(IV) Let $\boldsymbol{S_1}$ be the function defined in the table given and ***B*** be a block of 6-bits, then $\boldsymbol{S_1(B)}$ can be determined as follows: A number in the decimal range 0 to 3 (or binary 00 to 11) is used to define first and last bits of ***B*** represent in base 2. Let ***I*** denote that number. Let ***B*** represent the 4 bits which are in middle in base 2 a number that denotes the decimal range from 0 to 15 (binary 0000 to 1111). Let ***j*** be that number. The ***i***-th row and ***j***-th column represent the number by making use of look up table. The block which is of 4 bit is used to represent the number ranging from 0 to 15 uniquely.

Therefore, for the given input $\boldsymbol{BS_1(B)}$ of $\boldsymbol{S_1}$corresponds to the output. For example, for the given input block ***B*** = 011011 the first bit that represents "0" and the last bit that gives "1" giving 01 as the row. This is represented in row 1. The four bits to the middle are denoted as "1101", which describes the binary equivalent of decimal 13, so the column denotes column number 13. In row 1, and column13 appears 5. This helps in determining the output; 5 is binary 0101, so that the output is 0101. Hence $\boldsymbol{S_1}$(011011) = 0101.Thus the functions $\boldsymbol{S_1,\ldots,S_8}$can be described using the table as following:

Table 7: S_2Box.

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

Table 8: S_3 Box.

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

Table 9: S_4 Box.

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

Table 10: S_5 Box.

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

Table 11: S_6 Box.

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

Table 12: S_7 Box.

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

Table 13: S_8 Box.

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

The S-boxes that is used to represent the output is denoted by $f\,(R_{n-1},\ K_n)$. Furthermore, using the values of $L_{n-1}$and $R_{n-1}$for the values of n ranging from $n = 1, 2, \ldots 15$, we compute $R_n = L_{n-1} + f\ (R_{n-1}\ ,\ \ K_n)$. Furthermore towards end of the sixteenth round, we have the blocks L_{16} and R_{16}. The order comprising of the two blocks are then reversed into the 64-bit block $R_{16}\ L_{16}$.

(V) (Now the final permutation IP^{-1} to the 64-bit block $R_{16}L_{16}$, defined by the following Table 14, is applied:

Table 14: Final Permutation.

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

That is, bit 40 of the output containing the algorithm of the pre output block corresponds to its first bit, bit 8 as its second bit, and so on, pre output block that corresponds to bit 25 is the last bit of the output.

Step-3. Decryption Procedure of the Cipher Message:

Decryption mainly denotes the inverse of encryption, following the steps that are in synchronization as above, but order corresponding to the subkeys are applied are reversed.

(III) Illustration:

Let the message to be sent is: **HIMACHAL** and key used for encryption and decryption is **UNIVERSE**. Using ASCII, we write the given message in hexadecimal form and binary digit as follows:

Table 15.

H	I	M	A	C	H	A	L
48	69	6D	61	63	68	61	6C
0100 1000	0110 1001	0110 1101	0110 0001	0110 0011	0110 1000	0110 0001	0110 1100

Therefore, the given message becomes

M= 0100 1000 0110 1001 0110 1101 0110 0001 0110 0011 0110 1000 0110 0001 0110 1100,

which is 64 –bit plain text.

Step-1. Keys Generation:

We write the key Kin terms of binary digits as

K=0101 0101 0110 1110 0110 1001 0111 0110 0110 0101 0111 0010 0111 0011 0110 0101.

Using $PC1$ **Table**, the 64-bits key is permuted and it becomes 56-bits key as

K^* =0000000 0111111 1111111 1100110 0110101 0100110 1100000 1101001,

which is written in two parts as

C_0 =0000000 0111111 1111111 1100110

D_0=0110101 0100110 1100000 1101001.

Now, we create the pairs C_n and $D_n, 1 \le n \le 16$ from the previous pairs C_{n-1} and D_{n-1}. From the original pair of $C_0$and D_0 using the shifts given in Keys Generation of Data Encryption Standard Algorithm on page no. 12 and we obtain

C_1=0000000 1111111 1111111 0001100

D_1=1101010 10011001 1000001 1010010.

C_2= 0000001 1111111 1111110 0001100

D_2= 1010101 0011011 0000011 0100101.

C_3= 0000111 1111111 1111000 1100000

D_3= 010100 1101100 0001101 0010110.

C_4= 0011111 1111111 1100011 0000000

D_4=1010011 0110000 0110100 1011010.

C_5=1111111 1111111 0001100 0000000

D_5= 1001101 1000001 1010010 1101010.

C_6=1111111 1111100 0110000 0000011

D_6= 0110110 0000110 1001011 0101010.

C_7=1111111 1110001 1000000 0001111

D_7=1011000 0011010 0101101 0101001.

C_8=1111111 1000110 0000000 0111111

D_8=1100000 1101001 0110101 0100110.

C_9=1111111 0001100 0000000 1111111

D_9=1000001 1010010 1101010 1001101.

C_{10}=1111100 0110000 0000011 1111111

D_{10}=0000110 1001011 0101010 0110110.

C_{11}= 1110001 1000000 0001111 1111111

D_{11}= 0011010 0101101 0101001 1011000.

C_{12}= 10000110 0000000 0111111 1111111

D_{12}= 1101001 0110101 0100110 1100000.

C_{13}= 0011000 0000001 1111111 1111110

D_{13}= 0100101 1010101 0011011 0000011.

C_{14}= 1100000 0000111 1111111 1111000

D_{14}= 0010110 1010100 1101100 0001101.

C_{15}= 0000000 0011111 1111111 1100011

D_{15}=1011010 1010011 0110000 0110100.

C_{16}= 0000000 0111111 1111111 1000110

D_{16}= 0110101 0100110 1100000 1101001.

Now, to form the first key, we have

C_1D_1 = 0000000 1111111 1111111 00011000000000 1111111 1111111 0001100.

We use $PC2$ **Table** to change the given 56-bits key into 48-bits and corresponding key becomes

K_1 =111000 001011 011011 10010 010001 110000 011010 011111.

For the second key, we have

C_2D_2 =0000001 1111111 1111110 0011000 1010101 0011011 0000011 0100101.

Using$PC2$**Table**, the given 56-bits key is changed into 48-bits, so corresponding key becomes

K_2 =111100 001001 011001 111110 101000 001011 011100 010010.

For the third key, we have
C_3D_3= 0000111 1111111 1111000 1100000 1010100 1101100 0001101 0010110.
After using $\boldsymbol{PC2}$ **Table,** the given 56-bits key can be changed into 48-bits, so corresponding key becomes
K_3=111001 001101 101000 110010 011110 010010 011000 100110.
For the fourth key, we have
C_4D_4=0011111 1111111 1100011 0000000 1010011 0110000 0110100 1011010.
From $\boldsymbol{PC2}$ **Table,** the given 56-bits key can be changed into 48-bits, so corresponding key becomes
K_4=101001 101111 001101 110110 011111 000100 100011 011010.
For the fifth key, we have
C_5D_5=1111111 1111111 0001100 0000000 1001101 1000001 1010010 1101010.
After using $\boldsymbol{PC2}$ **table,** the given 56-bits key can be changed into 48-bits, so corresponding key becomes
K_5=101011 100101 011101 010011 000001 001111 000001 011110.
For the sixth key, we have
C_6D_6=1111111 1111100 0110000 0000011 0110110 0000110 0001011 0101010.
$\boldsymbol{PC2}$ **Table** changes the56-bits key into 48-bits, so corresponding key becomes
K_6= 011011 110101 001101 111001 101001 111011 010011 100000.
For the seventh key, we have
C_7D_7=1111111 1110001 1000000 0001111 1011000 0011010 0101101 0101001.
After using $\boldsymbol{PC2}$ **Table,** the given 56-bits key can be changed into 48-bits, so corresponding key becomes
K_7=100011 111101 000111 011001 101010 001000 111101 10001.
For the eighth key, we have
C_8D_8=1111111 1000110 0000000 0111111 1100000 1101001 0110101 0100110.
$\boldsymbol{PC2}$ **Table**changes the56-bits key into 48-bits, the corresponding key becomes
K_8= 000111 110100 101111 011011 000111 101100 111000 010110
For the ninth key, we have
C_9D_9=1111111 0001100 0000000 1111111 1000000 1010010 1101010 1001101.
Using $\boldsymbol{PC2}$ **Table,** the given 56-bits key can be changed into 48-bits, the corresponding key becomes
K_9= 001111 110100 101111 011001 100000 000001 100101 101110.
For the tenth key, we have
$C_{10}D_{10}$=11111100 0110000 0000011 1111111 0000110 1001011 0101010 0110110.
$\boldsymbol{PC2}$ **Table** changes the56-bits key into 48-bits, so corresponding key becomes
K_{10}= 000111 110011 100110 001101 110001 001011 101010 110100.
For the eleventh key, we have
$C_{11}D_{11}$=1110001 1000000 0001111 1111111 0011010 0101101 0101001 1011000.
After using $\boldsymbol{PC2}$ **Table,** the given 56-bits key can be changed into 48-bits, so corresponding key becomes
K_{11}=000110 110010 110011 011101 011100 010000 111011 111001.
For the twelfth key, we have
$C_{12}D_{12}$=1000011 0000000 0111111 1111111 1101001 0110101 0100110 1100000.
$\boldsymbol{PC2}$ **Table**changes the56-bits key into 48-bits, the corresponding key becomes
K_{12}=010111 010110 110010 101100 000110 111000 100000 011011.
For the thirteenth key, we have
$C_{13}D_{13}$=0011000 0000001 1111111 1111110 0100101 1010101 0011011 0000011.
Using $\boldsymbol{PC2}$ **Table,** the given 56-bits key can be changed into 48-bits, so corresponding key becomes
K_{13}=110100 101010 110110 101100 010011 110111 010100 110100.
For the fourteenth key, we have
$C_{14}D_{14}$=1100000 0000111 1111111 1111000 1100000 0000111 1111111 1111000.
$\boldsymbol{PC2}$ **Table**changes the56-bits key into 48-bits, so corresponding key becomes
K_{14}=110110 001010 111000 100111 001010 010010 100111 100100.
For the fifteenth key, we have
$C_{15}D_{15}$=0000000 0011111 1111111 1100011 1011010 1010011 0110000 0110100.
Using $\boldsymbol{PC2}$ **Table,** the given 56–bits key can be changed into 48-bits, so corresponding key becomes
K_{15}=111000 011011 111000 101110 111000 001100 100010 010111.
For the sixteenth key, we have
$C_{16}D_{16}$= 0000000 0111111 1111111 1000110 0110101 0100110 1100000 1101001.
After using $\boldsymbol{PC2}$ **Table,** the given 56–bits key can be changed into 48-bits, the corresponding key becomes
K_{16}=111000 001011 011010 101110 101100 111010 1000111 001000 .

Step-2.Encryption Procedure for the Original Message:

The initial permutation is applied making use of the Table-3 to the plaintext *M* given previously and get
M=1111 1111 0000 1000 0100 0101 0001 0000 0000 1111 1110 1010 0111 0001 0000Now, we further make use of 16 rounds for encryption. First, we divide *M* into two parts having 32 bits each as

L_0=1111 1111 0000 0000 1000 0100 0101 1110

and

R_0=0000 0000 1111 1110 1010 0111 0001 0000.

The following function is used for encryption in each round,

$L_n= R_{n-1}$,

$$R_n = L_{n-1} + f_k(R_{n-1}, K_n)$$

and

$$f_k(R_{n-1}, K_n)= K_n + \mathrm{E}(R_{n-1}),$$

where E(R_{n-1}) means expanding the size of $R_{n-1}$from 32-bits to 48-bits and K_n is already of 48-bits. Also, we use *S*-boxes under this function to compress the output ofK_n+ E(R_{n-1}), because we need only 32-bits to XORed with L_{n-1}, whereas output of K_n+ E(R_{n-1}) is of 48 bits so we compress it.

For first round, we take $n = 1$ and get,

L_1=R_0

and

$R_1 = L_0 + f(R_0, K_1)$.

Therefore,

L_1= 1111 1111 0000 0000 1000 0100 0101 1110.

Also,

R_0 =0000 00001111 1110 1010 0111 0001 0000

is of 32-bits using E-bit Table, we expand 32-bits into 48-bit and get

$E\ (R_0)$ =000000 000001 011111 111101 010100 001110 100010 100000.
Since K_1 is of 48-bits and after expansion R_0 becomes$E\ (R_0)$ with 48-bits. Therefore,
$K_1 + E\ (R_0)$ =111000 001010 000100 011011 000101 111110 111000 111111.
Let
$K_1 + E(R_0) = B_1B_2B_3B_4B_5B_6B_7B_8$,
where
B_1= 111000
B_2= 001010
B_3= 000100
B_4=011011
B_5=000101
B_6=111110
B_7=111000
B_8=111111.
Now, using *S*-boxes given on page no.16, 17, 18 and 19, we have
$S_1(B_1) = S_1$ (111000)
=2nd row and 12th column in S_1
=03
=0011,
$S_2(B_2) = S_2$ (001010)
= 0th row and 5th column in S_2
=11
=1011,
$S_3(B_3) = S_3$ (000100)
=0th row and 2nd column in S_3
=09
=1001,
$S_4(B_4) = S_4$ (011011)
=1st row and 13th column in S_4
=10
=1010,
$S_5(B_5) = S_5$ (000101)
= 1ST row and 2nd column in S_5
= 02
=0010,
$S_6(B_6) = S_6$ (111110)
=2nd row and 15th column in S_6
=06
=0110,
$S_7(B_7) = S_7$ (111000)
= 1st row and 12th column in S_7
=00
=0000,
$S_8(B_8) = S_8$ (111111)
=3rd row and 15th column in S_8
=11
=1011.
Therefore, the required output from S-boxes becomes $f(R_0, K_1)$ of 32- bits and have
$f\ (R_0, K_1)$=0011 1011 1001 1010 0010 0110 0000 1011.
Now,
$L_0 + f\ (R_0, K_1)$ =1100 0100 1001 1010 1010 0010 0101 0101.
Hence,
R_1=1100 0100 1001 1010 1010 0010 0101 0101
and
L_1=0000 0000 1111 1110 1010 0111 0001 0000.
For the second round, we take $n = 2$ and we get,
$L_2 = R_1$

and
$R_2 = L_1 + f(R_1, k_2)$.
From the previous round, we have
L_2=1100 0100 1001 1010 1010 0010 0101 0101.
Also,
R_2 =1100 0100 1001 1010 1010 0010 0101 0101
is of 32-bits. Using E-bit table, we expand 32-bits into 48-bits and get
$E(R_1)$=111000 001001 010011 110101 010100 000100 001010 101011
and
$K_2 + E(R_1)$=000100 000000 001010 001011 111100 001111 010110 111001.
Since$K_2 + E(R_1)$ is of 48-bits and L_1 is of 32-bits. Before adding these two, we firstly compress$K_2 + E(R_1)$ using *S*boxes given on page no. 16, 17, 18 and19 respectively.
Let
$K_2 + E(R_1)$=$B_1B_2B_3B_4B_5B_6B_7B_8$,
where
B_1= 000100
B_2= 000000
B_3= 001010
B_4= 001011
B_5= 111100
B_6= 001111
B_7 = 010110
B_8= 111001.
Since
$f\ (R_1, K_2)$=$S_i\ (B_i)$,$1 \le i \le 8$;
now,using *S*-boxes given on page no.16,17 and 18, we have
$S_1(B_1) = S_1$ (000100)
=0th row and 2nd column in S_1 =13
=1101.
$S_2(B_2) = S_2$ (000000)
= 0th row and 0th column in S_2
=15
=1111.
$S_3(B_3) = S_3$ (001010)
=0th row and 5th column in S_3
=03
=0011.
$S_4(B_4) = S_4$ (001011)
=1st row and 5th column in S_4
=15
=1111.
$S_5(B_5) = S_5$ (111100)
= 2nd row and 14th column in S_5
= 00
=0000.
$S_6(B_6)$= S_6 (001111)
=1st row and 7th column in S_6
=05
=0101.
$S_7(B_7) = S_7$ (010110)
= 0th row and 11th column in S
=07
=0111.
$S_8(B_8)$= S_8 (111001)
=3rd row and 12th column in S_8
=03
=0011.
So the required output from S-boxes becomes
$f(R_1, K_2)$=1101 1111 0011 1111 0000 0101 0111 0011.
Therefore,

$L_1 + f(R_1, K_2)$=1101 1111 1100 0001 1010 0010 0110 0011.
Hence,
R_2=1101 1111 1100 0001 1010 0010 0110 0011
and
L_2=1100 0100 1001 1010 1010 0010 0101 0101.
Using the similar procedure for $n = 3, \dots 15$, we get the required outputs.
Now, for the round $n = 16$, we have
$L_{16} = R_{15}$
and
$R_{16} = L_{15} + f(R_{15}, k_{16})$;
From previous round,
L_{16}=1011 0001 0110 1001 0000 0101 1100 0011.
From E-bit table, we expand 32-bits into 48-bits and get
R_{15} = 1011 0001 0110 1001 0000 0101 1100 0011,
then
$E(R_{15})$ =110110 100010 101101 010010 100000 000111 111000 000111.
Also,
$K_{16} + E(R_{15})$ =001110 101001 110111 111100 001100 111101 011111 001111.
$K_{16} + E(R_{15})$) is of 48-bits and L_{16} is of 32-bits. Adding these two, we firstly compressed $K_{16} + E(R_{15})$ using *S*-boxes given on page no.16, 17 and 18 respectively.
Let
$K_{16} + E(R_{15})$ =$B_1B_2B_3B_4B_5B_6B_7B_8$,
where
B_1= 001110
B_2= 101001
B_3= 110111
B_4= 111100
B_5= 001100
B_6= 111101
B_7= 011111
B_8= 001111.
Since,
$f(R_{15}, K_{16}) = S_i(B_i), 1 \le i \le 8$;
now,using *S*-boxes given on page no.16,17, 18 and 19, we have
$S_1(B_1) = S_1$(001110)
=0th row and 7th column in S_1
=08
=1000.
$S_2(B_2) = S_2$(101001)
= 3rd row and 4th column in S_2
=03
=0011.
$S_3(B_3) = S_3$(110111)
=3rd row and 11rd column in S_3
=03
=0011.
$S_4(B_4) = S_4$ (111100)
=2nd row and 14th column in S_4
=08
=1000.
$S_5(B_5) = S_5$ (001100)
= 0th row and 6th column in S_5
= 11
=1011.
$S_6(B_6) = S_6$ (111101)
=3rd row and 14th column in S_6
=08
=1000.
$S_7(B_7) = S_7$ (011111)
=1st row and 15th column in S_7
=06
=0110.
$S_8(B_8) = S_8$ (001111)
=1st row and 7th column in S_8
=04
=0100.
So the required output from S-boxes becomes
$f(R_{15}, K_{16})$ =1000 0011 0011 1000 1011 1000 0110 0100.
Therefore,
L_{15}+ $f(R_{15}, K_{16})$ =0011 1001 0101 1100 1110 0101 1000 0001.
Hence,
R_{16}= 0011 1001 0101 1100 1110 0101 1000 0001
and
L_{16}= 1011 0001 0110 1001 0000 0101 1100 0011.
So, the corresponding cipher text after 16 round becomes
$L_{16}R_{16}$ =1011 0001 0110 1001 0000 0101 1100 0011 1001 0101 1100 1110 0101 1000 0001.
Now, using the Final Permutation Table -13, the corresponding cipher text becomes
C=11011111 00000001 00101100 10110000 11100000 11011000 00111001 01001011.
In hexadecimal it is written as
13 15 01 02 12 11 00 14 00 13 08 39 04 11.
Therefore, the cipher text is
DC3NAKSOHSTXDC2DC1NULLDC3BS9EOTDC1
which is sent through public channels.

Step-3. Decryption Procedure of the Cipher Message:

Decryption algorithm is the inverse of the encryption algorithm. The output of encryption algorithm becomes input in the decryption algorithm. Therefore, the given cipher message becomes
M=1101 1111 0000 0001 0010 1100 1011 0000 1110 0000 1101 1000 0011 1001 0100 1011.
Using Initial Permutation Table, M becomes
M = 1011 0001 0110 1001 0000 0101 1100 0011 0011 1001 0101 1100 1110 0101 1000 0001.
We use the functions
$$R_{n-1} = L_n$$
and
$L_{n-1} = R_n + f(L_n, k_n)$.
For the first round, we take $n = 16$ and get
$$R_{15} = L_{16}$$
and
$$L_{15} = R_{16} + f(L_{16}, k_{16}).$$
We divide M into two half parts, which are given by
L_{16}=1011 0001 0110 1001 0000 0101 1100 0011
and
R_{16}= 0011 1001 0101 1100 1110 0101 1000 0001.
Since
$$R_{15} = L_{16},$$
therefore,
R_{15} = 1011 0001 0110 1001 0000 0011 1100 0011.
We use same key
K_{16} = 111000 001011 011010 101110 101100 111010 100111 001000
as used for encryption of the original message.
Now, using E-bit Table, we have
$E(L_{16})$ = 110110 100010 101101 010010 100000 000111 111000 000111.
Therefore,
$K_{16} + E(L_{16})$ = 001110 101001 110111 111100 001100 111101 011111 001111,

which is same as in fifteenth round of encryption. Using S-boxes as in encryption of original message, we have

$f(L_{16}, K_{16})$ =1000 0011 0011 1000 1011 1000 0110 0100

and

$R_{16} + f(L_{16}, K_{16})$ =1011 1010 0110 0100 0101 1101 1110 0101.

Therefore,

L_{15}=1011 1010 0110 0100 0101 1101 1110 0101

and

R_{15}= 1011 0001 0110 1001 0000 0011 1100 0011.

For the second round, we take $n = 15$ and get

$$R_{14} = L_{15}$$

and

L_{14}=$R_{15} + f(L_{15}, K_{15})$.

From the previous round, we have

R_{14}=1011 1010 0110 0100 0101 1101 1110 0101.

Here the key is

K_{15} =111000 011011 111000 101110 111000 001100 100010 010111.

From E-bit Table, we have

$E(L_{15})$ =110111 110100 001100 001000 001011 111011 111100 001011.

Therefore ,

$K_{15} + E(L_{15})$ =001111 101111 110100 100110 110011 110111 011110 011100,

which is same as in fourteenth round of encryption. Using S-boxes as in encryption, we have

$f(L_{15}, K_{15})$ =1010 0010 0010 0000 1111 0111 0001 1000

and

R_{15}+ $f(L_{15}, K_{15})$= 0001 0011 0100 1001 1111 0100 1101 1111.

Hence

L_{14}= 0001 0011 0100 1001 1111 0100 1101 1111

and

R_{14}=1011 1010 0110 0100 0101 1101 1110 0101.

Using the similar procedure for $n = 14, 13 \dots 2$, we get the required outputs.

For the round sixteenth $n = 1$, we get

$$R_0 = L_1$$

and

L_0= $R_1 + f(L_1, k_1)$.

From the previous round, we have

R_0 =0000 0000 1111 1110 1010 0111 0001 0000.

Here the key is

K_1 =111000 001011 011011 100110 010001 110000 011010 011111.

Using E-bit Table, we have

L_1 =0000 0000 1111 1110 1010 0111 0001 0000

and

$E(L_1) = 000000$ 000001 011111 111101 010100 001110 100010 100000.

Therefore,

K_1+$E(L_1)$ =111000 001010 000100 011011 000101 111110 111000 111111

which is same as in first round of encryption. We use S-boxes as in encryption and get

$f(L_1, K_1)$=0011 1011 1001 1010 0010 0110 0000 1011

and

$R_1 + f(L_1, K_1)$ =1111 1111 0000 0000 1000 0100 0101 1110.

Therefore,

R_0 = 0000 0000 1111 1110 1010 0111 0001 0000

and

L_0=1111 1111 0000 0000 1000 0100 0101 1110.

So, the required output is written as

$M = L_0R_0 =$ 1111 1111 0000 0000 1000 0100 0101 1110 0000 0000 1111 1110 1010 0111 0001 0000.

Using Final Permutation Table, we get required message as

M =0100 1000 0110 1001 0110 1101 0110 0001 0110 0011 0110 1000 0110 0001 0110 1100.

Now, we write the above message in hexadecimals as

48 69 6D 61 63 68 61 6C.

The required plaintext after using ASCII is

$\boldsymbol{M} =$ **HIMACHAL.**

III. CONCLUSION

Data Encryption Standard is the important technique of processing the data in Symmetric key cryptography. We discussed in detail all the levels of Data Encryption Standards and focused on the mathematical algorithm which provides the security to the data.

REFERENCES

[1]. Buchmann, J. A.(2004). Introduction to Cryptography, New York.

[2]. Carlet, C., Goubin, L., Prouff, E., Quisquater, M.,& Rivain M. (2012). Higher-Order Masking Schemes for S-Boxes. Fast Software Encryption FSE 2012, *Lecture Notes in Computer Science, 7549*, 366-384.

[3]. Coppersmith, D. (1994). The Data Encryption Standard (DES) and its strength against attacks.*IBM J. of Research and Development, 38*(3), 243-250.

[4]. Data Encryption Standard, Federal Information Processing Standard (FIPS) Publication 46, National Bureau of Standards, U.S. Department of Commerce, Washington D.C. (1977).

[5]. Grabbe J. Orlin. The DES Algorithm Illustrated, 'Laissez' Faire City Times, *2*(8). Homepage:http://www. aci. net/kalliste/homepage. Html.

[6]. Kenekayoro Patrick T. (2010). The Data Encryption Standard thirty four years later: An overview, *African J. of Maths. & Computer Science Research, 3*(10), 267-269.

[7]. Menezes, A. J., Paul, C. Van Oorschot,& Vanstone, S. A. (1997). Handbook of Applied Cryptography, CRC Press, Boca Raton.

[8]. Meyer C. H.,& Matyas S. M., Cryptography: A New Dimension in Computer Data Security. John Wiley & Sons, New York, (1982).

[9]. Phan, R. C. W. (2007). Reducing the exhaustive key search of the Data Encryption Standard (DES), Computer Standards & Interfaces, *29*, 528-530.

[10]. Schenier B. (1996). Applied Cryptography, Second Edition, John Wiley &Sons, New York.

[11] Sharma, A. K. (2018). Content-Based Filtering in Movie Recommendation. *International Journal of Electrical, Electronics and Computer Engineering*, *7*(2): 106-109.

[12] Sharma, A. K. (2019). Design and Mathematical Structure of Cryptographic Hash Function SHA-512. *International Journal of Theoretical & Applied Sciences, 11*(2): 41-47.

[13] Sharma, A. K. (2019). Safety Application in Android. *International Journal on Emerging Technologies, 10*(1): 234-00.

[14] Sharma, A. K., & Badoga, N. K. (2020). Digital Signatures Using RSA Public Key Cryptosystem Scheme. *International Journal of Theoretical & Applied Sciences, 12*(1): 37-42.

[15]. Stalling, W. (2012). Cryptography and Network Security, Pearson,

[16]. Stinson, D. R.(1995). Cryptography: Theory and Practice, CRC Press, Boca Raton.

How to cite this article: Sharma, A. K., and Badoga, N. K. (2020). Data Encryption Standard Algorithm in Symmetric Key Cryptography over Finite Field F_2. *International Journal on Emerging Technologies, 11*(5): 705–712.