Analytical Approach for Data Encryption Standard Algorithm

https://doi.org/10.3991/ijim.v17i14.38641

Obaida M. Al-hazaimeh[1](✉), Moyawiah A. Al-Shannaq[2], Mohammed J. Bawaneh[1], Khalid M.O. Nahar[2]
[1] Al-Balqa Applied University, Irbid, Jordan
[2] Yarmouk University, Irbid, Jordan
dr_obaida@bau.edu.jo

Abstract—Although it was first developed and studied in the late 1970s and early 1977s, the Data Encryption Standard (DES) algorithm has grown in popularity. There are two causes for this occurrence. First, the DES algorithm's complex mathematical structure allows it to serve as the theoretical foundation for a wide variety of applications. Second, the encryption technique works quite well in practice for a variety of applications when implemented correctly. In this paper, we undertake a thorough and practical review of the theoretical aspects of this sort of encryption algorithm and demonstrate how they have been implemented by executing multiple encryption configurations.

Keywords—data encryption standard, cryptography, S-box, bit rotation, symmetric cipher

1 Introduction

The Data Encryption Standard (DES) is a standard technique for securing computer and telecommunication data. The National Bureau of Standards (now the National Institute of Standards and Technology) first adopted this standard in 1977 as FIPS Pub 46 [1, 2]. DES is a block cipher of the Feistel type that operates on 64-bit data blocks with a 56-bit key [3, 4].Feistel ciphers use numerous rounds to decipher a block of bits by independently processing its left and right halves. To be invertible, a Feistel cipher just requires that the function (f) used to operate on the half-blocks of data bits be invertible, which is an intriguing property in and of itself. Because it executes both substitutions and permutations, the function f in the Data Encryption Algorithm is a product cipher.The Data Encryption Standard (DES) is an example of a symmetric block cipher [5, 6]. Each 64-bit block of plaintext is converted into ciphertext using a 56-bit key.The 56-bit key used to encipher the text is the same key used to decrypt it. The only variation between encryption and decryption is in the formation of sub-keys, therefore the key and algorithm are the same in both cases. Permutations, initial and inverse initial, and 16 comparable rounds of permutations, summing, and bit-wise manipulations are used in DES's processing of input blocks. The initial permutation merely passes input bits to different processing locations. The second bit, for example, is routed to location 33, and bit 55 is routed to place two. This rerouting is clearly explained in FIPS Pub-46. The inverse initial permutation is simply the opposite of the initial permutation. The block is processed in 16 iterations between the initial and inverse initial permutations. The DES algorithm's block diagram is presented in Figure 1 [5, 7-9].

This paper is structured as follows. In Section 2 we review the theory of DES algorithm with simple example. In section 3 illustrates the evaluation of the DES algorithm. Section 4 concludes the conclusion.

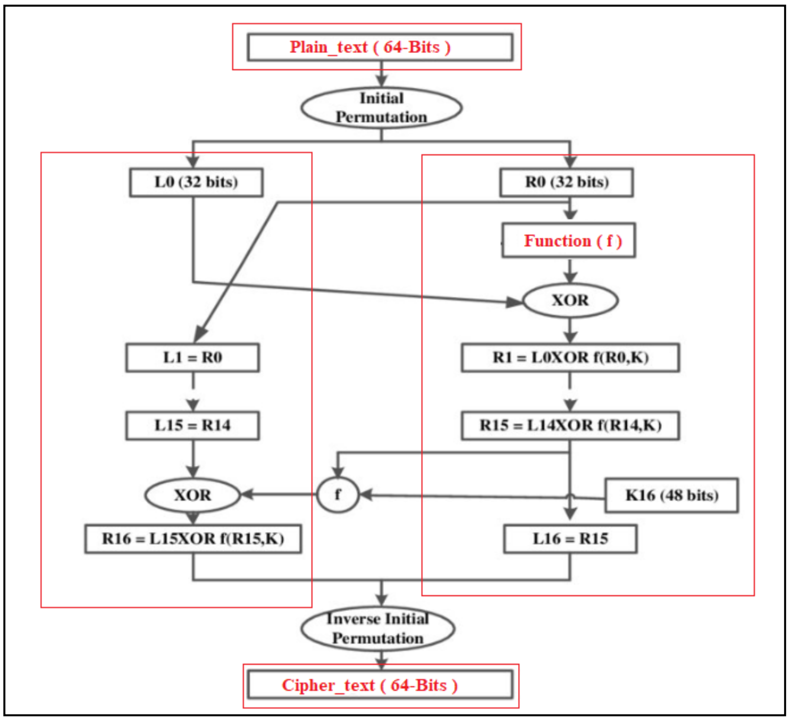

Fig. 1. Block diagram of DES algorithm

2 DES Processes – Example

The DES algorithm encrypts messages in blocks of 64 bits, which is equivalent to 16 hexadecimal digits. The "keys" that DES employs to encrypt data are reportedly 16 hexadecimal digits long, or 64 bits. The DES algorithm uses a 56-bit key, but discards every eighth bit as noise. In any event, 64-bits (i.e., 16 Hexadecimal digits) is the round number based on which DES is structured. DES algorithm is based on the fundamental parts: Subkeys generation and encryption process[10, 11].These parts are explained in the following subsections bellow.

2.1 Sub keys generator

This phase make a 16 sub-keys, each with a length of 48-bits. The Sub-key generation process based on sequential steps:

Let K be the Hexadecimal key K = 133457799BBCDFF1.

1. Convert the *K* to the binary key based on Figure 2 as illustrated in Table 1 and Figure 3.

Table 1. Key representation - Binary

Range	Group	HEX	Binary
[1-2]	1	13	00010011
[3-4]	2	34	00110100
[5-6]	3	57	01010111
[7-8]	4	79	01111001
[9-10]	5	9B	10011011
[11-12]	6	BC	10111100
[13-14]	7	DF	11011111
[15-16]	8	F1	11110001

```python
my_hexdata = "133457799BBCDFF1"

scale = 16 ## equals to hexadecimal

num_of_bits = 8

bin(int(my_hexdata, scale))[2:].zfill(num_of_bits)
```

Fig. 2. Convert key to binary sequence

Key = 0001001100110100010101110111100110011011101111001101111111110001

35	34	33	32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
0	0	1	1	0	0	1	1	1	1	0	1	1	1	0	1	0	1	0	0	0	1	0	1	1	0	0	1	1	0	0	1	0	0	0
Given Key- 64-Bits						64	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36
						1	0	0	0	1	1	1	1	1	1	1	1	1	0	1	1	0	0	1	1	1	1	0	1	1	1	0	1	1

Fig. 3. 64-Bits key length

2. Key permutation to 56-Bits length

The 64-bit key is permuted according to the Table 2 to generate a 56-Bit. As the first number in the table, "57" indicates that the 57th bit of the original key is the new starting point for the permuted key. The second bit of the permuted key is the original key's 49th bit. Bit 4 of the original key is now the final bit of the permuted key. Remember that the permuted key only contains 56-bits of the original key as shown in Figure 4.

15	23	31	39	47	55	63	36	44	52	60	3	11	19	27	35	43	51	59	2	10	18	26	34	42	50	58	1	9	17	25	33	41	49	57
0	1	0	1	0	1	0	1	1	1	1	0	1	0	1	0	1	0	1	0	0	1	1	0	0	1	1	0	0	0	0	1	1	1	1
Permuted Key-56-Bits														4	12	20	28	5	13	21	29	37	45	53	61	6	14	22	30	38	46	54	62	7
														1	1	1	1	0	0	0	1	1	1	1	0	0	1	1	0	0	1	1	0	1

Fig. 4. 56-Bits permuted key

Table 2. Index permutation

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

3. Split the permuted key into 2 blocks (i.e., C_0 and D_0), each block 28-Bits as shown in Figure 5.

C0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
	1	1	1	1	0	0	0	0	1	1	0	0	1	1	0	0	1	0	1	0	1	0	1	0	1	1	1	1

D0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
	0	1	0	1	0	1	0	1	0	1	1	0	0	1	1	0	0	1	1	1	1	0	0	0	1	1	1	1

Fig. 5. 28-Bits each block

4. Shift each block based on the shift left value to generate 16-Subkeys as shown in Figure 6 and 7.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	Shift Left By
C0	1	1	1	1	0	0	0	0	1	1	0	0	1	1	0	0	1	0	1	0	1	0	1	0	1	1	1	1	
C1	1	1	1	0	0	0	0	1	1	0	0	1	1	0	0	1	0	1	0	1	0	1	0	1	1	1	1	1	1
C2	1	1	0	0	0	0	1	1	0	0	1	1	0	0	1	0	1	0	1	0	1	0	1	1	1	1	1	1	1
C3	0	0	0	0	1	1	0	0	1	1	0	0	1	0	1	0	1	0	1	0	1	1	1	1	1	1	1	1	2
C4	0	0	1	1	0	0	1	1	0	0	1	0	1	0	1	0	1	0	1	1	1	1	1	1	1	1	0	0	2
C5	1	1	0	0	1	1	0	0	1	0	1	0	1	0	1	0	1	1	1	1	1	1	1	1	0	0	0	0	2
C6	0	0	1	1	0	0	1	0	1	0	1	0	1	0	1	1	1	1	1	1	1	1	0	0	0	0	1	1	2
C7	1	1	0	0	1	0	1	0	1	0	1	0	1	1	1	1	1	1	1	1	0	0	0	0	1	1	0	0	2
C8	0	0	1	0	1	0	1	0	1	0	1	1	1	1	1	1	1	1	0	0	0	0	1	1	0	0	1	1	2
C9	0	1	0	1	0	1	0	1	0	1	1	1	1	1	1	1	1	0	0	0	0	1	1	0	0	1	1	0	1
C10	0	1	0	1	0	1	0	1	1	1	1	1	1	1	1	0	0	0	0	1	1	0	0	1	1	0	0	1	2
C11	0	1	0	1	0	1	1	1	1	1	1	1	1	0	0	0	0	1	1	0	0	1	1	0	0	1	0	1	2
C12	0	1	0	1	1	1	1	1	1	1	1	0	0	0	0	1	1	0	0	1	1	0	0	1	0	1	0	1	2
C13	0	1	1	1	1	1	1	1	1	0	0	0	0	1	1	0	0	1	1	0	0	1	0	1	0	1	0	1	2
C14	1	1	1	1	1	1	1	0	0	0	0	1	1	0	0	1	1	0	0	1	0	1	0	1	0	1	0	1	2
C15	1	1	1	1	1	0	0	0	0	1	1	0	0	1	1	0	0	1	0	1	0	1	0	1	0	1	1	1	2
C16	1	1	1	1	0	0	0	0	1	1	0	0	1	1	0	0	1	0	1	0	1	0	1	0	1	1	1	1	1

Fig. 6. Result of the shift left operation -Right block

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	Shift Left By
D0	0	1	0	1	0	1	0	1	0	1	1	0	0	1	1	0	0	1	1	1	1	0	0	0	1	1	1	1	
D1	1	0	1	0	1	0	1	0	1	1	0	0	1	1	0	0	1	1	1	1	0	0	0	1	1	1	1	0	1
D2	0	1	0	1	0	1	0	1	1	0	0	1	1	0	0	1	1	1	1	0	0	0	1	1	1	1	0	1	1
D3	0	1	0	1	0	1	1	0	0	1	1	0	0	1	1	1	1	0	0	0	1	1	1	1	0	1	0	1	2
D4	0	1	0	1	1	0	0	1	1	0	0	1	1	1	1	0	0	0	1	1	1	1	0	1	0	1	0	1	2
D5	0	1	1	0	0	1	1	0	0	1	1	1	1	0	0	0	1	1	1	1	0	1	0	1	0	1	0	1	2
D6	1	0	0	1	1	0	0	1	1	1	1	0	0	0	1	1	1	1	0	1	0	1	0	1	0	1	0	1	2
D7	0	1	1	0	0	1	1	1	1	0	0	0	1	1	1	1	0	1	0	1	0	1	0	1	0	1	1	0	2
D8	1	0	0	1	1	1	1	0	0	0	1	1	1	1	0	1	0	1	0	1	0	1	0	1	1	0	0	1	2
D9	0	0	1	1	1	1	0	0	0	1	1	1	1	0	1	0	1	0	1	0	1	0	1	1	0	0	1	1	1
D10	1	1	1	1	0	0	0	1	1	1	1	0	1	0	1	0	1	0	1	0	1	1	0	0	1	1	0	0	2
D11	1	1	0	0	0	1	1	1	1	0	1	0	1	0	1	0	1	0	1	1	0	0	1	1	0	0	1	1	2
D12	0	0	0	1	1	1	1	0	1	0	1	0	1	0	1	0	1	1	0	0	1	1	0	0	1	1	1	1	2
D13	0	1	1	1	1	0	1	0	1	0	1	0	1	0	1	1	0	0	1	1	0	0	1	1	1	1	0	0	2
D14	1	1	1	0	1	0	1	0	1	0	1	0	1	1	0	0	1	1	0	0	1	1	1	1	0	0	0	1	2
D15	1	0	1	0	1	0	1	0	1	0	1	1	0	0	1	1	0	0	1	1	1	1	0	0	0	1	1	1	2
D16	0	1	0	1	0	1	0	1	0	1	1	0	0	1	1	0	0	1	1	1	1	0	0	0	1	1	1	1	1

Fig. 7. Result of the shift left operation - Left block

5. Recombine C_n and D_n to generate 16- Sub keys each sub key size is 56-Bits as shown in Figure 8.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35
1	1	1	0	0	0	0	1	1	0	0	1	1	0	0	1	0	1	0	1	0	1	0	1	1	1	1	1	1	0	1	0	1	0	1
36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	**Key No.1 – 56-Bits**													
0	1	1	0	0	1	1	0	0	1	1	1	1	0	0	0	1	1	1	1	0														
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35
1	1	0	0	0	0	1	1	0	0	1	1	0	0	1	0	1	0	1	0	1	0	1	1	1	1	1	1	0	1	0	1	0	1	0
36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	**Key No.2– 56-Bits**													
1	1	0	0	1	1	0	0	1	1	1	1	0	0	0	1	1	1	1	0	1														
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35
0	0	0	0	1	1	0	0	1	1	0	0	1	0	1	0	1	0	1	0	1	1	1	1	1	1	1	1	0	1	0	1	0	1	1
36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	**Key No.3– 56-Bits**													
0	0	1	1	0	0	1	1	1	1	0	0	0	1	1	1	1	0	1	0	1														
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35
0	0	1	1	0	0	1	1	0	0	1	0	1	0	1	0	1	0	1	1	1	1	1	1	1	1	0	0	0	1	0	1	1	0	0
36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	**Key No.4– 56-Bits**													
1	1	0	0	1	1	1	1	0	0	0	1	1	1	1	0	1	0	1	0	1														

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35
1	1	0	0	1	1	0	0	1	0	1	0	1	0	1	0	1	1	1	1	1	1	1	1	0	0	0	0	0	1	1	0	0	1	1
36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	**Key No.5– 56-Bits**													
0	0	1	1	1	1	0	0	0	1	1	1	1	0	1	0	1	0	1	0	1														
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35
0	0	1	1	0	0	1	0	1	0	1	0	1	0	1	1	1	1	1	1	1	1	0	0	0	0	1	1	1	0	0	1	1	0	0
36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	**Key No.6– 56-Bits**													
1	1	1	1	0	0	0	1	1	1	1	0	1	0	1	0	1	0	1	0	1														
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35
1	1	0	0	1	0	1	0	1	0	1	0	1	1	1	1	1	1	1	1	0	0	0	0	1	1	0	0	0	1	1	0	0	1	1
36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	**Key No. 7– 56-Bits**													
1	1	0	0	0	1	1	1	1	0	1	0	1	0	1	0	1	0	1	1	0														
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35
0	0	1	0	1	0	1	0	1	0	1	1	1	1	1	1	1	1	0	0	0	0	1	1	0	0	1	1	1	0	0	1	1	1	1
36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	**Key No.8– 56-Bits**													
0	0	0	1	1	1	1	0	1	0	1	0	1	0	1	0	1	1	0	0	1														

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35
0	1	0	1	0	1	0	1	0	1	1	1	1	1	1	1	1	0	0	0	0	1	1	0	0	1	1	0	0	0	1	1	1	1	0
36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	**Key No.9– 56-Bits**													
0	0	1	1	1	1	0	1	0	1	0	1	0	1	0	1	1	0	0	1	1														
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35
0	1	0	1	0	1	0	1	1	1	1	1	1	1	1	0	0	0	0	1	1	0	0	1	1	0	0	1	1	1	1	1	0	0	0
36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	**Key No.10– 56-Bits**													
1	1	1	1	0	1	0	1	0	1	0	1	0	1	1	0	0	1	1	0	0														
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35
0	1	0	1	0	1	1	1	1	1	1	1	1	0	0	0	0	1	1	0	0	1	1	0	0	1	0	1	1	1	0	0	0	1	1
36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	**Key No.11– 56-Bits**													
1	1	0	1	0	1	0	1	0	1	0	1	1	0	0	1	1	0	0	1	1														
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35
0	1	0	1	1	1	1	1	1	1	1	0	0	0	0	1	1	0	0	1	1	0	0	1	0	1	0	1	0	0	0	1	1	1	1
36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	**Key No.12– 56-Bits**													
0	1	0	1	0	1	0	1	0	1	1	0	0	1	1	0	0	1	1	1	1														

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35
0	1	1	1	1	1	1	1	1	0	0	0	0	1	1	0	0	1	1	0	0	1	0	1	0	1	0	1	0	1	1	1	1	0	1
36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	**Key No.13– 56-Bits**													
0	1	0	1	0	1	0	1	1	0	0	1	1	0	0	1	1	1	1	0	0														
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35
1	1	1	1	1	1	1	0	0	0	0	1	1	0	0	1	1	0	0	1	0	1	0	1	0	1	0	1	1	1	1	0	1	0	1
36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	**Key No.14– 56-Bits**													
0	1	0	1	0	1	1	0	0	1	1	0	0	1	1	1	1	0	0	0	1														
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35
1	1	1	1	1	0	0	0	0	1	1	0	0	1	1	0	0	1	0	1	0	1	0	1	0	1	1	1	1	0	1	0	1	0	1
36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	**Key No.15– 56-Bits**													
0	1	0	1	1	0	0	1	1	0	0	1	1	1	1	0	0	0	1	1	1														
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35
1	1	1	1	0	0	0	0	1	1	0	0	1	1	0	0	1	0	1	0	1	0	1	0	1	1	1	1	0	1	0	1	0	1	0
36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	**Key No.16– 56-Bits**													
1	0	1	1	0	0	1	1	0	0	1	1	1	1	0	0	0	1	1	1	1														

Fig. 8. Sub keys – 56-Bits length

6. Key permutation to 48-Bits length

The 56-bit key is permuted according to the Table 3 to generate a 48-bit as shown in Figure 9.

Table 3. Index permutation to 48-bits length

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

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35
	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
Key No.1 – 48-Bits	0	0	0	1	1	0	1	1	0	0	0	0	0	0	1	0	1	1	1	0	1	1	1	1	1	1	1	1	1	1	0	0	0	1	1
Key No.2 – 48-Bits	0	1	1	1	1	0	0	1	1	0	1	0	1	1	1	0	1	1	0	1	1	0	0	1	1	1	0	1	1	0	1	1	1	1	0
Key No.3 – 48-Bits	0	1	0	1	0	1	0	1	1	1	1	1	1	1	0	0	1	0	0	0	1	0	1	0	0	1	0	0	0	0	1	0	1	1	0
Key No.4 – 48-Bits	0	1	1	1	0	0	1	0	1	0	1	0	1	1	0	1	1	1	0	1	0	1	1	0	1	1	0	1	1	0	1	1	0	0	1
Key No.5 – 48-Bits	0	1	1	1	1	1	0	0	1	1	1	0	1	1	0	0	0	0	0	0	0	1	1	1	1	1	1	0	1	0	1	1	0	1	0
Key No.6 – 48-Bits	0	1	1	0	0	0	1	1	1	0	1	0	0	1	0	1	0	0	1	1	1	1	1	0	0	1	0	1	0	0	0	0	0	1	1
Key No.7 – 48-Bits	1	1	1	0	1	1	0	0	1	0	0	0	0	1	0	0	1	0	1	1	0	1	1	1	1	1	1	1	0	1	1	0	0	0	0
Key No.8 – 48-Bits	1	1	1	1	0	1	1	1	1	0	0	0	1	0	1	0	0	0	1	1	1	0	1	0	1	1	0	0	0	0	0	1	0	0	1
Key No.9 – 48-Bits	1	1	1	0	0	0	0	0	1	1	0	1	1	0	1	1	1	1	1	0	1	0	1	1	1	1	1	0	1	1	0	1	1	1	1
Key No.10 – 48-Bits	1	0	1	1	0	0	0	1	1	1	1	1	0	0	1	1	0	1	0	0	0	1	1	1	1	0	1	1	1	0	1	0	0	1	0
Key No.11 – 48-Bits	0	0	1	0	0	0	0	1	0	1	0	1	1	1	1	1	1	1	0	1	0	0	1	1	1	1	0	1	1	1	1	0	1	1	0
Key No.12– 48-Bits	0	1	1	1	0	1	0	1	0	1	1	1	0	0	0	1	1	1	1	1	0	1	0	1	1	0	0	1	0	1	0	0	0	1	1
Key No.13 – 48-Bits	1	0	0	1	0	1	1	1	1	1	0	0	0	1	0	1	1	1	0	1	0	0	0	1	1	1	1	1	1	0	1	0	1	0	1
Key No.14 – 48-Bits	0	1	0	1	1	1	1	1	0	1	0	0	0	0	1	1	1	0	1	1	0	1	1	1	1	1	1	1	0	0	1	0	1	1	1
Key No.15 – 48-Bits	1	0	1	1	1	1	1	1	1	0	0	1	0	0	0	1	1	0	0	0	1	1	0	1	0	0	1	1	1	1	0	1	0	0	1
Key No.16 – 48-Bits	1	1	0	0	1	0	1	1	0	0	1	1	1	1	0	1	1	0	0	0	1	0	1	1	0	0	0	0	1	1	1	0	0	0	0

	36	37	38	39	40	41	42	43	44	45	46	47	48
	48	44	49	39	56	34	53	46	42	50	36	29	32
Key No.1 – 48-Bits	1	0	0	0	0	0	1	1	1	0	0	1	0
Key No.2 – 48-Bits	0	1	0	0	1	1	1	1	0	0	1	0	1
Key No.3 – 48-Bits	0	1	1	1	1	1	0	0	1	1	0	0	1
Key No.4 – 48-Bits	1	0	1	0	1	0	0	0	1	1	1	0	1
Key No.5 – 48-Bits	1	0	0	1	1	1	0	1	0	1	0	0	0
Key No.6 – 48-Bits	1	1	0	1	1	0	0	1	0	1	1	1	1
Key No.7 – 48-Bits	1	1	0	0	0	1	0	1	1	1	1	0	0
Key No.8 – 48-Bits	1	1	0	1	1	1	1	1	1	1	0	1	1
Key No.9 – 48-Bits	0	0	1	1	1	1	0	0	0	0	0	0	1
Key No.10 – 48-Bits	0	0	1	1	0	0	1	0	0	1	1	1	1
Key No.11 – 48-Bits	1	0	0	1	1	1	0	0	0	0	1	1	0
Key No.12– 48-Bits	0	0	1	1	1	1	1	1	0	1	0	0	1
Key No.13 – 48-Bits	1	1	0	1	0	0	1	0	0	0	0	0	1
Key No.14 – 48-Bits	0	0	1	1	1	0	0	1	1	1	0	1	0
Key No.15 – 48-Bits	1	1	1	1	1	0	0	0	0	1	0	1	0
Key No.16 – 48-Bits	1	0	1	1	1	1	1	1	1	0	1	0	1

Fig. 9. 16-Sub keys – 48-Bits key length for each sub key

As soon as this phase is completed, we have generated the sub keys K_n, for $1 \leq n \leq 16$, by applying the permutation structure based on Table 3 to each of the joined C_nD_n pairs. Now let's take a look at the actual message (i.e., Plaintext) by applying the second phase (i.e., Encryption process).

2.2 Encryption process

Now let's take a look at the actual message (i.e., plaintext) by applying the second phase (i.e., Encryption process). The encryption process based on sequential steps:
Let plaintext be the Hexadecimal M = 0123456789ABCDEF.

1. Convert the Plaintext to the binary key based on Figure 10 as illustrated in Table 4 and Figure 11.

Table 4. Key binary representation

Range	Group	HEX	Binary
[1-2]	1	01	00000001
[3-4]	3	23	00100011
[5-6]	5	45	01000101
[7-8]	7	67	01100111
[9-10]	9	89	10001001
[11-12]	11	AB	10101011
[13-14]	13	CD	11001101
[15-16]	15	EF	11101111

```python
my_hexdata = "0123456789ABCDEF"

scale = 16 ## equals to hexadecimal

num_of_bits = 8

bin(int(my_hexdata, scale))[2:].zfill(num_of_bits)
```

Fig. 10. Convert message to binary sequence

Message is 0000000100100011010001010110011110001001101010111100110111101111

35	34	33	32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
0	0	1	1	1	1	0	0	1	1	0	1	0	1	0	0	0	1	0	1	1	0	0	0	1	0	0	1	0	0	0	0	0	0	0
Plaintext – 64-Bits						64	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36
						1	1	1	1	0	1	1	1	1	0	1	1	0	0	1	1	1	1	0	1	0	1	0	1	1	0	0	1	0

Fig. 11. Plaintext- 64-Bits length

2. Plain-text initial permutation
Initially, the 6-bits that make up the message data M are shuffled around. Each entry in Table 5 indicates how the bits have been rearranged from their original order. M's 58th bit is now the first bit. As a result, bit 50 of M is now bit 2. As for M, the seventh bit is the very last one as shown in Figure 12.

Table 5. Initial permutation

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

35	34	33	32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
41	**49**	**57**	**8**	**16**	**24**	**32**	**40**	**48**	**56**	**64**	**6**	**14**	**22**	**30**	**38**	**46**	**54**	**62**	**4**	**12**	**20**	**28**	**36**	**44**	**53**	**60**	**2**	**10**	**18**	**26**	**34**	**42**	**50**	**58**
1	1	1	1	1	1	1	1	1	1	1	0	0	1	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	1
Permutated Plaintext-64-Bits						64	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36
						7	**15**	**23**	**31**	**39**	**47**	**55**	**63**	**5**	**13**	**21**	**29**	**37**	**45**	**53**	**61**	**3**	**11**	**19**	**27**	**35**	**43**	**51**	**59**	**1**	**9**	**17**	**25**	**33**
						0	1	0	1	0	1	0	1	0	0	0	0	1	1	1	1	0	1	0	1	0	1	0	1	0	0	0	0	1

Fig. 12. Result of the initial permutation

3. L_n and R_n

Here, DES split the permuted message (See Figure 12) into two halves, each of which consists of 32-bits (i.e., Left and Right) as shown in Figure 13.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	**Left L_0**
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**	
1	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	1	0	0	1	1	1	1	1	1	1	1	
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	**Right R_0**
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**	
1	1	1	1	0	0	0	0	1	0	1	0	1	0	1	0	1	1	1	1	0	0	0	0	1	0	1	0	1	0	1	0	

Fig. 13. Left and right parts

Now, perform DES structure to generate L_0 to L_{16} and R_0 to R_{16} based on the following formula:

$L_n=R_{n-1}$

$R_n=L_{n-1}$ XOR $F(R_{n-1}, K_n)$

L_1	$L1=R_{1-1}$	$L_1=R_0$
R_1	$R_1=L_{1-1}$ XOR $F(R_{1-1}, K_1)$	$R1=L_0$ XOR $F(R_0, K_1)$
L_2	$L_2=R_{2-1}$	$L_2=R_1$
R_2	$R_1=L_{2-1}$ XOR $F(R_{2-1}, K_2)$	$R_2=L_1$ XOR $F(R_1, K_2)$
L_3	$L_3=R_{3-1}$	$L_3=R_2$
R_3	$R_3=L_{3-1}$ XOR $F(R_{3-1}, K_3)$	$R_3=L_2$ XOR $F(R_2, K_3)$
.	.	.
.	.	.
.	.	.
L_{15}	$L_{15}=R_{15-1}$	$L_{15}=R_{14}$
R_{15}	$R_{15}=L_{15-1}$ XOR $F(R_{15-1}, K_{15})$	$R_{15}=L_{15}$ XOR $F(R_{14}, K_{15})$
L_{16}	$L_{16}=R_{16-1}$	$L_{16}=R_{15}$
R_{16}	$R_{16}=L_{16-1}$ XOR $F(R_{16-1}, K_{16})$	$R_{16}=L_{15}$ XOR $F(R_{15}, K_{16})$

The Implementation process for 2-rounds to generate L_1 and R_1 as follow:

Figure 14 displayed L_1. To find R1, we get R0 = 32-Bits and K1 = 48- Bits. DES algorithm expands R0 to be 48-Bits based on Table 6 (i.e., Bit-selection). Table 6, repeats some of the bits in R_{n-1} to expand 32-bit to 48-bits as shown in Figure 15.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	L_1
1	1	1	1	0	0	0	0	1	0	1	0	1	0	1	0	1	1	1	1	0	0	0	0	1	0	1	0	1	0	1	0	

Fig. 14. L_1 32-bits length

Table 6. Bit-selection

32	1	2	3	4	5
8	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

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	Expanded (R0) 48-Bits
32	**1**	**2**	**3**	**4**	**5**	**8**	**5**	**6**	**7**	**8**	**9**	**8**	**9**	**10**	**11**	**12**	**13**	**12**	**13**	**14**	**15**	**16**	**17**	
0	1	1	1	1	0	1	0	0	0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	
25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	
16	**17**	**18**	**19**	**20**	**21**	**20**	**21**	**22**	**23**	**24**	**25**	**24**	**25**	**26**	**27**	**28**	**29**	**28**	**29**	**30**	**31**	**32**	**1**	
0	1	1	1	1	0	1	0	0	0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	

Fig. 15. Expanded R_0 to 48-bits length

In Figure 16, the expanded R0 is XORed with the Sub key No.1 (See Figure 9)

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	Expanded (R0) 48-Bits
32	**1**	**2**	**3**	**4**	**5**	**8**	**5**	**6**	**7**	**8**	**9**	**8**	**9**	**10**	**11**	**12**	**13**	**12**	**13**	**14**	**15**	**16**	**17**	
0	1	1	1	1	0	1	0	0	0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	
25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	
16	**17**	**18**	**19**	**20**	**21**	**20**	**21**	**22**	**23**	**24**	**25**	**24**	**25**	**26**	**27**	**28**	**29**	**28**	**29**	**30**	**31**	**32**	**1**	
0	1	1	1	1	0	1	0	0	0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	Key 1 48-Bits
0	0	0	1	1	0	1	1	0	0	0	0	0	0	1	0	1	1	1	0	1	1	1	1	
25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	
1	1	1	1	1	1	0	0	0	1	1	1	0	0	0	0	0	1	1	1	0	0	1	0	

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	[Expanded (R0)] XOR [Key 1]
0	1	1	0	0	0	0	1	0	0	0	1	0	1	1	1	1	0	1	1	1	0	1	0	
25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	
1	0	0	0	0	1	1	0	0	1	1	0	0	1	0	1	0	0	1	0	0	1	1	1	

Fig. 16. Result of XOR operation

As can be seen in Figure 17, the XORed data has been partitioned into 8 groups, each of which consists of 6-bits.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	Groups Selection
0	1	1	0	0	0	0	1	0	0	0	1	0	1	1	1	1	0	1	1	1	0	1	0	
Group No. 1						Group No. 2						Group No. 3						Group No. 4						
25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	
1	0	0	0	0	1	1	0	0	1	1	0	0	1	0	1	0	0	1	0	0	1	1	1	
Group No. 5						Group No. 6						Group No. 7						Group No. 8						

Fig. 17. Group selection

DES uses six-bit groups as addresses in "S-boxes" as shown in Figure 18. Each six-bit group gives an S-box address. A 4-bit number is at that address. This 4-bit number replaces 6 bits. The eight groups of 6-bits are turned into eight groups of 4-bits (S-box outputs) for 32 bits. S-box value is determined as shown in Table 7.

No.	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

S1-Box– Group No. 1

No.	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
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

S2-Box – Group No. 2

No.	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
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

S3-Box– Group No. 3

No.	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
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

S4-Box– Group No. 4

No.	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
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

S5-Box– Group No. 5

No.	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
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	141	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

S6-Box– Group No. 6

No.	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
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
S7-Box– Group No. 7																

No.	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	13	2	8	4	6	15	11	1	10	9	3	141	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
S8-Box– Group No. 8																

Fig. 18. S-Box for each group

Table 7. S-box utilization

Group No.	Binary Representation	Binary	Decimal	S-Box Intersection	S-Box
Group No.1	011000			5	(See Figure 18) S1 – Box-Group No. 1
Row	First & Last-Bit (2-Bits)	00	0		
Column	In Between (4-Bits)	1100	12		
Group No.2	010001			12	(See Figure 18) S2 – Box-Group No. 2
Row	First & Last-Bit (2-Bits)	01	1		
Column	In Between (4-Bits)	1000	8		
Group No.3	011110			8	(See Figure 16)S3 – Box-Group No. 3
Row	First & Last-Bit (2-Bits)	00	0		
Column	In Between (4-Bits)	1111	15		
Group No.4	111010			2	(See Figure 18) S4 – Box-Group No. 4
Row	First & Last-Bit (2-Bits)	10	2		
Column	In Between (4-Bits)	1101	13		
Group No.5	100001			11	(See Figure 18) S5 – Box-Group No. 5
Row	First & Last-Bit (2-Bits)	11	3		
Column	In Between (4-Bits)	0000	0		
Group No.6	100110			5	(See Figure 18) S6 – Box-Group No. 6
Row	First & Last-Bit (2-Bits)	10	2		
Column	In Between (4-Bits)	0011	3		
Group No.7	010100			9	(See Figure 18) S7– Box-Group No. 7
Row	First & Last-Bit (2-Bits)	00	0		
Column	In Between (4-Bits)	1010	10		
Group No.8	100111			7	(See Figure 18) S8 – Box-Group No. 8
Row	First & Last-Bit (2-Bits)	11	3		
Column	In Between (4-Bits)	0011	3		

Figure 19 displays the result derived from the preceding table (Table 7).

Decimal	5	12	8	2	11	5	9	7
Binary (4-bits)	0101	1100	1000	0010	1011	0101	1001	0111

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

Fig. 19. Binary representation of the obtained result

4. Function (f)

The final step in calculating f is to permute the S-box output using Table 8 to achieve the final value of f as shown in Figure 20.

Table 8. Index permutation

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

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
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**
0	0	1	0	0	0	1	1	0	1	0	0	1	0	1	0	1	0	1	0	1	0	0	1	1	0	1	1	1	0	1	1

Fig. 20. F(R_0, K_1)

R_1=L_0 XOR F(R_0, K_1), we got the result in Figure 21 to represent R_1.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	
0	0	1	0	0	0	1	1	0	1	0	0	1	0	1	0	1	0	1	0	1	0	0	1	1	0	1	1	1	0	1	1	**F(R_0, K_1)**
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	
1	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	1	0	0	1	1	1	1	1	1	1	1	**L_0**
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	
1	1	1	0	1	1	1	1	0	1	0	0	1	0	1	0	0	1	1	0	0	1	0	1	0	1	0	0	0	1	0	0	**R_1**

Fig. 21. R_1

Repeating the preceding steps yields L_{16} and R_{16}as indicated in Figure 22.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	
0	1	0	0	0	0	1	1	0	1	0	0	0	0	1	0	0	0	1	1	0	0	1	0	0	0	1	1	0	1	0	0	**L_{16}**
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	
0	0	0	0	1	0	1	0	0	1	0	0	1	1	0	0	1	1	0	1	1	0	0	1	1	0	0	1	0	1	0	1	**R_{16}**

Fig. 22. L_{16} and R_{16}

5. Re-combine L_{16} and R_{16}

We have the blocks L_{16} and R_{16} at the end of the sixteenth round. The two blocks are then reversed into the 64-bit block (i.e., $R_{16}L_{16}$) as shown in Figure 23.

35	34	33	32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
0	1	0	1	0	1	0	1	0	0	1	1	0	0	1	1	0	1	1	0	0	1	1	0	0	1	0	0	1	0	1	0	0	0	0
$R_{16}L_{16}$						64	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36
						0	0	1	0	1	1	0	0	0	1	0	0	1	1	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0

Fig. 23. $R_{16}L_{16}$

Finally, utilize Table 9's for final Permutation as shown in Table 9 and Figure 24.

Table 9. Initial permutation - Inverse

40	8	48	16	56	24	64	32
39	7	47	15	55	23	64	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

35	34	33	32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
44	**4**	**36**	**29**	**61**	**21**	**53**	**13**	**45**	**5**	**37**	**30**	**62**	**22**	**54**	**14**	**46**	**6**	**38**	**31**	**63**	**23**	**55**	**15**	**47**	**7**	**39**	**32**	**64**	**24**	**56**	**16**	**48**	**8**	**40**
0	0	0	0	0	1	0	1	0	1	0	1	1	0	0	1	0	0	0	0	0	0	1	0	1	1	1	1	0	1	0	0	0	0	1
						64	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36
Cipher text-64-Bits						**25**	**57**	**17**	**49**	**9**	**41**	**1**	**33**	**26**	**58**	**18**	**50**	**10**	**42**	**2**	**34**	**27**	**59**	**19**	**51**	**11**	**43**	**3**	**35**	**28**	**60**	**20**	**52**	**12**
						1	0	1	0	0	0	0	0	0	0	1	0	1	1	0	1	0	1	0	1	0	0	0	0	1	1	1	1	0

Fig. 24. Binary representation of the Cipher text -64-Bits

If we express the result as a Hexadecimal, we get the value as shown in Table 10 for both plain and cipher texts.

Table 10. HEX data representation for Plain and Cipher texts

DATA	HEXADECIMAL Number	SIZE
Plaintext	0123456789ABCDEF	16 * 4 = 64- Bits
Cipher text	85E813540F0AB405	16 * 4 = 64- Bits

To make it clear, we have outlined how DES algorithm operates to convert plain text to cipher text in Figures 25, 26, and 27.

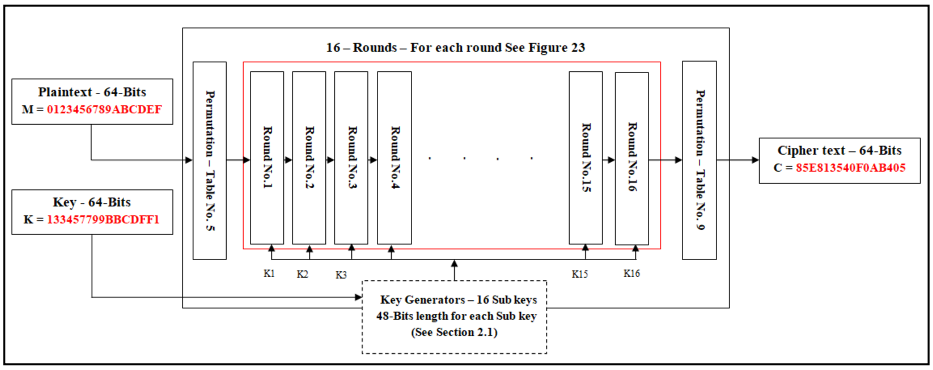

Fig. 25. Steps of DES algorithm

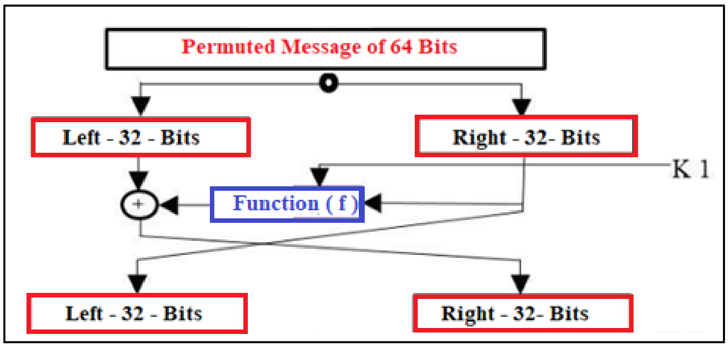

Fig. 26. DES-For each round

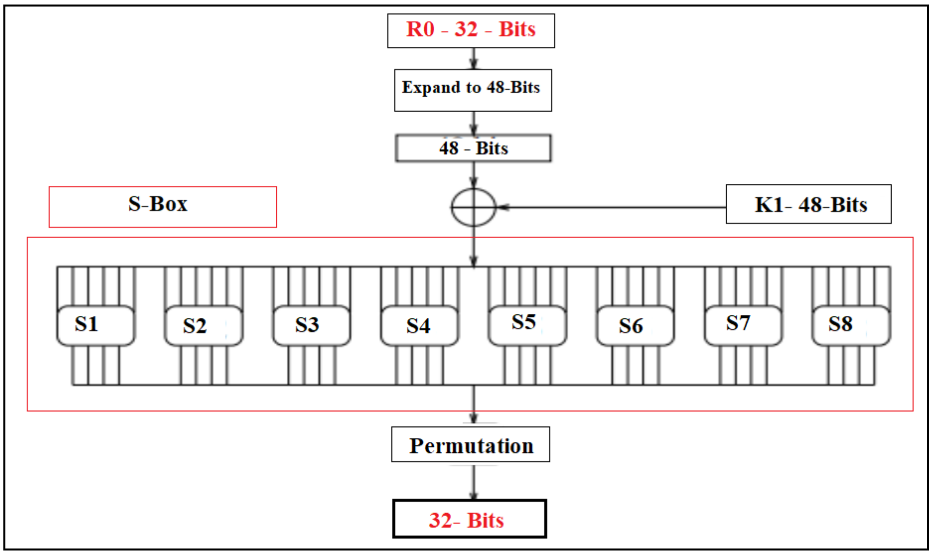

Fig. 27. Function F(R_{n-1}, K_n)

3 Evaluation

Data Encryption Standard (i.e., DES) is an obsolete symmetric key encryption technique[12]. It was implemented by government entities in 1977 to safeguard sensitive information and was officially decommissioned in 2005 [13, 14].The initial specification was developed by IBM's researchers in the early 1970s. Afterwards, in 1977, it became an official Federal Information Processing Standard (i.e., FIPS) for the encryption of commercial and sensitive but unclassified government computer data by the U.S. National Bureau of Standards, now known as the National Institute of Standards and Technology, or NIST[6, 15, 16].The United States government first allowed the public release of DES, an encryption algorithm. In doing so, it secured its rapid adoption by sectors like the financial sector, which required robust encryption. Due to its ease of use, DES was also implemented in many embedded devices, such as smart cards, SIM cards, modems, and routers [17-19].

Brute-force attacks, in which each possible key is tried in turn until the correct one is discovered, are the most fundamental way to break any cipher[20]. The number of potential keys, and hence the viability of this form of attack, is directly proportional to the key length as shown in Table 11 [7, 16, 21].A maximum of 256 or nearly 72 quadrillion, attempts would be needed to find the correct key, given that the effective DES key length is 56 bits. This is insufficient to prevent modern computers from breaking through DES-protected data using brute force[22]. Through the early to middle 1990s, DES was still the de facto standard for secure data transmission. While this may seem like a long time, in 1998, a computer created by the Electronic Frontier Foundation (i.e., EFF) deciphered a DES-encoded communication in just 56 hours. The next year, EFF reduced the decryption time to 22 hours by harnessing the power of thousands of networked computers. The extended version of DES, known as Triple-DES (i.e., TDES or 3-DES), will be used in government and finance for many years to come. The differences and similarities between DES and Triple-DES are shown in Table 12 [14, 23-29].

Table 11. Total Key-Search Time: an Average Estimate

Key size - Bits	Quantity of alternate keys	Decryption time - 1 /µs
168	$2^{168} = 3.7 \text{ x}10^{50}$	2^{167}µs = $5.9 \text{ x } 10^{36}$ years
128	$2^{128} = 3.4 \text{ x}10^{38}$	2^{127}µs = $5.4 \text{ x}10^{24}$ years
56	$2^{56} = 7.2 \text{ x } 10^{16}$	2^{55}µs = 1142 years
32	$2^{32} = 4.3 \text{ x } 10^{9}$	2^{31}µs = 35.8minutes

Table 12. DES and Triple-DES - Comparison

Feature	DES	Triple-DES
Created by / Year	IBM - 1975	IBM - 1978
Key size	56 - Bits	(168 or 112) - Bits
Number of rounds	16	48
Number of Sub keys	16	48
Key generations	Permute Shift	Permute Shift
Block size	64 – Bits	64 – Bits
Mathematical operations	S-Boxes- Fixed, XOR	S-Boxes- Fixed, XOR
Cipher type	Block cipher - Symmetric	Block cipher - Symmetric
Attack	Broken in 1998 - Brute-force	No known Attack
Security level	Not Secure Enough	Adequate Security
Speed	Slow	Very Slow

4 Conclusion

In this paper we have attempted to present the theory of Data Encryption Standard (i.e., DES) algorithm from the simplest to the most sophisticated concept. It has been our purposes to focus on the practical explanation of the basic mathematics and how DES algorithm it could be implemented in practice in real world system because a correct comprehension of the encryption algorithm leads to an understanding of the pros and cons, which allows for modernization and the development of a new encryption algorithm that operates more efficiently.

5 References

[1] O. C. Abikoye, A. D. Haruna, A. Abubakar, N. O. Akande, and E. O. Asani, "Modified advanced encryption standard algorithm for information security," *Symmetry,* vol. 11, p. 1484, 2019. https://doi.org/10.3390/sym11121484

[2] H. B. Acla and B. D. Gerardo, "Security Analysis of Lightweight Encryption based on Advanced Encryption Standard for Wireless Sensor Networks," in *2019 IEEE 6th International Conference on Engineering Technologies and Applied Sciences (ICETAS)*, 2019, pp. 1-6. https://doi.org/10.3390/sym11121484

[3] O. M. A. AI-Hazaimeh, "Design of a new block cipher algorithm," *Network and Complex Systems,* vol. 3, pp. 1-5, 2013.

[4] O. Al-Hazaimeh, N. Alhindawi, S. M. Hayajneh, and A. Almomani, "HANON chaotic map-based new digital image encryption algorithm," *MAGNT Research Report,* vol. 2, pp. 261-266, 2014.

[5] R. Bhanot and R. Hans, "A review and comparative analysis of various encryption algorithms," *International Journal of Security and Its Applications,* vol. 9, pp. 289-306, 2015. https://doi.org/10.3390/sym11121484

[6] O. M. Al-hazaimeh, "A novel encryption scheme for digital image-based on one dimensional logistic map," *Computer and Information Science,* vol. 7, p. 65, 2014. https://doi.org/10.5539/cis.v7n4p65

[7] T. Hagras, D. Salama, and H. Youness, "Anti-attacks encryption algorithm based on DNA computing and data encryption standard," *Alexandria Engineering Journal,* vol. 61, pp. 11651-11662, 2022. https://doi.org/10.5539/cis.v7n4p65

[8] S.-J. Han, H.-S. Oh, and J. Park, "The improved data encryption standard (DES) algorithm," in *Proceedings of ISSSTA'95 International Symposium on Spread Spectrum Techniques and Applications*, 1996, pp. 1310-1314.

[9] M. Khan and N. Munir, "A novel image encryption technique based on generalized advanced encryption standard based on field of any characteristic," *Wireless personal communications,* vol. 109, pp. 849-867, 2019. https://doi.org/10.5539/cis.v7n4p65

[10] M. E. Smid and D. K. Branstad, "Data encryption standard: past and future," *Proceedings of the IEEE,* vol. 76, pp. 550-559, 1988. https://doi.org/10.1109/5.4441

[11] Z. Yingbing and L. Yongzhen, "The design and implementation of a symmetric encryption algorithm based on DES," in *2014 IEEE 5th International Conference on Software Engineering and Service Science*, 2014, pp. 517-520. https://doi.org/10.1109/ICSESS.2014.6933619

[12] O. Laia and E. Zamzami, "Analysis of Combination Algorithm Data Encryption Standard (DES) and Blum-Blum-Shub (BBS)," in *Journal of Physics: Conference Series*, 2021, p. 012017. https://doi.org/10.1109/ICSESS.2014.6933619

[13] M. F. Mushtaq, S. Jamel, A. H. Disina, Z. A. Pindar, N. S. A. Shakir, and M. M. Deris, "A survey on the cryptographic encryption algorithms," *International Journal of Advanced Computer Science and Applications,* vol. 8, 2017. https://doi.org/10.14569/IJACSA.2017.081141

[14] C. Paar and J. Pelzl, "The data encryption standard (DES) and alternatives," in *Understanding Cryptography*, ed: Springer, 2010, pp. 55-86. https://doi.org/10.14569/IJACSA.2017.081141
[15] O. M. Al-Hazaimeh, "A new speech encryption algorithm based on dual shuffling Hénon chaotic map," *International Journal of Electrical and Computer Engineering (IJECE),* vol. 11, pp. 2203-2210, 2021. https://doi.org/10.14569/IJACSA.2017.081141
[16] O. M. Al-Hazaimeh, N. Alhindawi, and N. A. Otoum, "A novel video encryption algorithm-based on speaker voice as the public key," in *2014 IEEE International Conference on Control Science and Systems Engineering*, 2014, pp. 180-184. https://doi.org/10.1109/CCSSE.2014.7224533
[17] S. K. Rakeshkumar, "Performance analysis of data encryption standard algorithm & proposed data encryption standard algorithm," *International Journal of Engineering Research and Development,* vol. 7, pp. 11-20, 2013.
[18] D. Sensarma and S. S. Sarma, "GMDES: A Graph Based Modified Data Encryption Standard Algorithm With Enhanced Security‖," *International Journal of Research in Engineering and Technology,* vol. 3, pp. 653-660, 2014. https://doi.org/10.15623/ijret.2014.0303121
[19] N. Tahat, A. A. Tahat, M. Abu-Dalu, R. B. Albadarneh, A. E. Abdallah, and O. M. Al-Hazaimeh, "A new RSA public key encryption scheme with chaotic maps," *International Journal of Electrical & Computer Engineering (2088-8708),* vol. 10, 2020. https://doi.org/10.11591/ijece.v10i2.pp1430-1437
[20] N. Tahat, A. Alomari, A. Al–Freedi, O. M. Al-Hazaimeh, and M. F. Al–Jamal, "An efficient identity-based cryptographic model for Chebyhev chaotic map and integer factoring based cryptosystem," *Journal of Applied Security Research,* vol. 14, pp. 257-269, 2019. https://doi.org/10.11591/ijece.v10i2.pp1430-1437
[21] O. M. A. Al-hazaimeh, "New cryptographic algorithms for enhancing security of voice data," Universiti Utara Malaysia, 2010.
[22] M. Abdelwahab, M. Rizk, and H. Slim, "Extracting algebraic equations of the 2-key simplified 3-DES for algebraic cryptanalysis," *Journal of Engineering Science and Military Technologies,* vol. 3, pp. 1-13, 2019. https://doi.org/10.21608/ejmtc.2019.9737.1105
[23] L. Wang and G. Jiang, "The design of 3-DES encryption system using optimizing keys," in *2019 China-Qatar International Workshop on Artificial Intelligence and Applications to Intelligent Manufacturing (AIAIM)*, 2019, pp. 56-58. https://doi.org/10.1109/AIAIM.2019.8632786
[24] O. M. A. Al-Hazaimeh, "A new approach for complex encrypting and decrypting data," *International Journal of Computer Networks & Communications,* vol. 5, p. 95, 2013. https://doi.org/10.1109/AIAIM.2019.8632786
[25] R. Sharma and R. Sharma, "A Novel Approach Using 3-Des Algorithm Against Cryptographic Attacks," *Think India Journal,* vol. 22, pp. 8130-8139, 2019. https://doi.org/10.26643/think-india.v22i3.8314
[26] S. Zeebaree, "DES encryption and decryption algorithm implementation based on FPGA," *Indones. J. Electr. Eng. Comput. Sci,* vol. 18, pp. 774-781, 2020. https://doi.org/10.11591/ijeecs.v18.i2.pp774-781
[27] M. A. Dar, A. Askar, D. Alyahya, and S. A. Bhat, "Lightweight and Secure Elliptical Curve Cryptography (ECC) Key Exchange for Mobile Phones," *International Journal of Interactive Mobile Technologies,* vol. 15, 2021. https://doi.org/10.3991/ijim.v15i23.26337
[28] I. M. ALattar and A. M. S. Rahma, "A New Block Cipher Algorithm Using Magic Square of Order Five and Galois Field Arithmetic with Dynamic Size Block," *International Jour-*

nal of Interactive Mobile Technologies, vol. 15, 2021. https://doi.org/10.3991/ijim.v15i16.24187

[29] A. S. Jamil and A. M. S. Rahma, "Cyber Security for Medical Image Encryption using Circular Blockchain Technology Based on Modify DES Algorithm," *International Journal of Online & Biomedical Engineering,* vol. 19, 2023. https://doi.org/10.3991/ijoe.v19i03.37569

6 Authors

Obaida M. Al-Hazaimeh earned a BSc in Computer Science from Jordan's Applied Science University in 2004 and an MSc in Computer Science from Malaysia's University Science Malaysia in 2006. In 2010, he earned a PhD in Network Security (Cryptography) from Malaysia. He is a Full professor at Al-Balqa Applied University's Department of Computer Science and Information Technology. Cryptology, image processing, machine learning, and chaos theory are among his primary research interests. He has published around 57 papers in international refereed publications as an author or co-author. He can be contacted at email: dr_obaida@bau.edu.jo.

Moyawiah A. Al-Shannaq is an associate Professor of Computer Sciences in the Faculty of Information Technology and Computer Science, Yarmouk University. I received my MSc and BSc in Computer Sciences from Yarmouk University, Jordan. I received my PhD in Computer Sciences from Kent State University, Ohio, USA (2015). His main research interests are stenography, machine learning, algorithmic graph and hypergraph theory, and natural language processing. He can be contacted at email: moyawiah.s@yu.edu.jo.

Mohammed J. Bawaneh is an associate Professor in the Department of Computer Science and Information Technology. He received his PhD in computer information systems. His main research interests are stenography and machine learning. Now, he is a lecturer at Al-Balqa Applied University – Al-huson University College, Jordan. He can be contacted at email: dr_mjab@bau.edu.jo.

Khalid M.O. Nahar is an associate Professor in the Department of Computer Sciences-Faculty of IT, Yarmouk University, Irbid-Jordan. He received his BS and MS in Computer Sciences from Yarmouk University in Jordan, in 1992 and 2005, respectively. He was awarded a full scholarship to continue his PhD in Computer Sciences and Engineering from King Fahd University of Petroleum and Minerals (KFUPM), KSA. In 2013, he completed his PhD and started his job as an Assistant Professor at Tabuk University, KSA for two years. In 2015, he backs to Yarmouk University and from now he is the chairman of training and development department. He can be contacted at email: Khalids@yu.edu.jo.

Article submitted 2023-02-07. Resubmitted 2023-04-01. Final acceptance 2023-04-01. Final version published as submitted by the authors.