Mathematical Formulation of DES Algorithm.

Abstract

Today modern Technology like wireless network helps us to connect instantly with people anywhere. And main object the Security of Wireless network is faced by today's world. For that cryptography plays a big role to provide security to the wireless network. And main, numerous encryption calculations are accessible to secure the information. This paper objective is ordinarily utilized symmetric encryption calculation which is DES Algorithm.

1. INTRODUCTION

Cryptography is a method of protecting information and communications through the use of codes, so that only those for whom the information is intended can read and process it. Or Cryptography is an art of conveying messages in coded form which is understood only by the intended recipient. The recipient in turn decodes to read the message. The transfer of data through public network with security issues can be protected with cryptography. In Cryptography have several standard symmetric and asymmetric algorithms which are highly secured and time tested.

2. TYPES OF CRYPTOGRAPHY

Cryptography is the following two main types.

1. Symmetric algorithm
2. Asymmetric algorithm

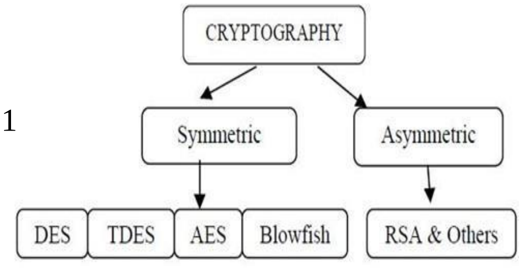
Symmetric Algorithm.

This is the simplest kind of encryption that involves only one secret key to cipher and decipher information. Symmetrical encryption is an old and best-known technique. Symmetric algorithm is a type of algorithm which we have a single key which is kept secret among the sender and the recipient is used so that no unauthorized person can use the data, which is to be transferred safe.

Asymmetric Algorithm.

Asymmetrical encryption is also known as public key cryptography, which is a relatively new method, compared to symmetric encryption. Asymmetric algorithm is a type of algorithm which we have a public key and private key are used where public is known to everyone whereas the private key is only known to the recipient of the message. This encryption is considered more secure when compared with symmetric algorithm. When speed is compared symmetric algorithm works faster and very safe.

Figure:1



3. PROPOSED WORK

The DES has the following steps involved.

DES is a ***block cipher***--meaning it operates on plaintext blocks of a given size (64-bits) and returns ciphertext blocks of the same size. Thus DES results in a ***permutation*** among the 2^64 (read this as: "2 to the 64th power") possible arrangements of 64 bits, each of which may be either

a. 1st step 64 bit plain text is taken as input and initial permutation is done on the input by re-arranging the bits to get the permuted input.

b. 2nd step is the next step involves 16 rounds of the same function along with permutation and substitution.

c. 3rd step is the 16th output contains 64 bits as a result of function of input plain text and key.

d. 4Th step is the output of left and right side are swapped producing the preoutis.

e. 5th step is the preoutis have gone through IP, i.e. Opposite of initial permutation to produce 64 bit cipher text.as showen paper.

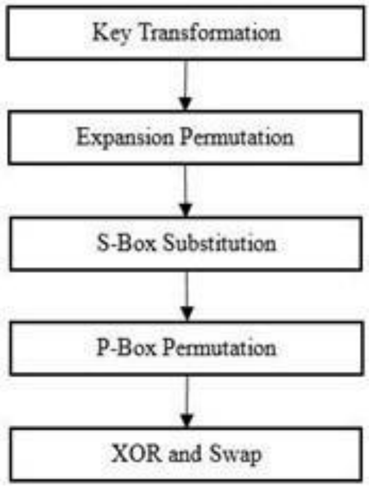


Figure 2

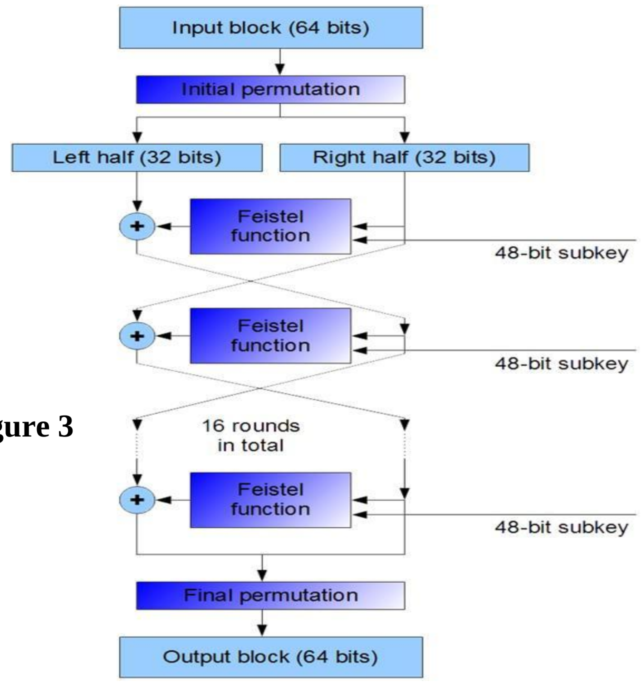


Figure 3

3.2 DES-Feistel function

The following figure shows that.
Which of the following is the definition of Feistel?
Function?
A cryptographic function that splits blocks of data intotwo parts; one of the most influential developments in symmetric block ciphers.

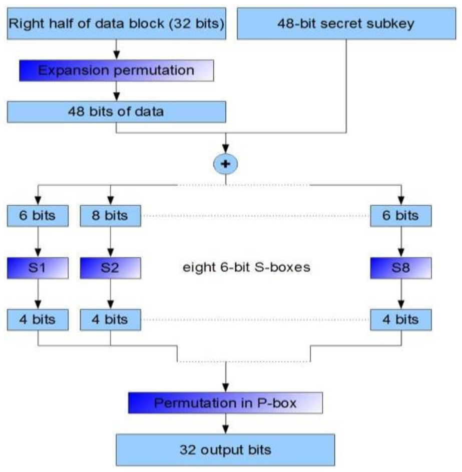


Figure 4

The encryption process uses the Feistel **structure** consisting multiple rounds of processing of the plaintext, each round consisting of a “substitution” step followed by a permutation step. The input block to each round is divided into two halves that can be denoted as L and R for the left half and the right half. As show in fig 4 and after fig 5.

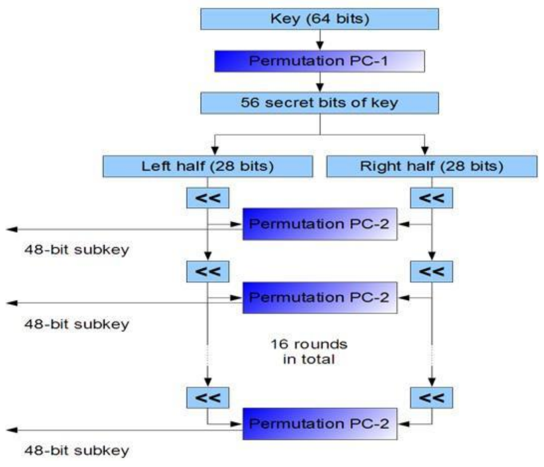


Figure 5

3.5 MATHEMATICAL PROCEDURE

1. The presentation of permutations are in the table form only to understand easily. The data which are input are vectors and are not matrices.

2. The tables of permutation are read row by row from left to right and from top to bottom.

3. The bits which are permuted are read as follows.

4. The resultant first output is the bit from the input blocks whose position taken from the first row and the first column of the table.

5. The resultant second output is the bit from the input block whose position is taken from the first row and the second column of the table.

6. The resultant last output is the bit from the input block whose position is taken from the first row and the last column of the table.

3.4 Initial Permutation

Table show, It is done on every block of input data in the beginning stage of encryption.

Detail:

Initial Permutation is the process of arranging all shuffling each bit of original plaintext block with only other random bit of same plaintext manage block.
As the table show that first bit of original plaintext block replace with 48th bit of original plaintext block the 2nd bit replace with 57th bit of original plaintext message. This process called jugglery of bit position of plaintext block which is applied to all original plain text block is a sequence.
After initial permeation the 64 bit plaintext block get divide in to two half LPT(32-bit)

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

3.5 Permutation Key PC-1

The following Table show, the from 64-bit key only 56 bits are selected. The key is then divided as left half and right half. Bit shifting is done on every part of it . (Every eight bit can be used for parity control which is excluded from encryption. The 64-bit key is permuted according to the following table, **PC-1**. 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. The 4th bit of the original key is the last bit of the permuted key. Note only 56 bits of the original key appear in the permuted key.

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
		Right half				
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

3.6 Permutation Expansion

The table shows Every round Feistel function is initiated by expansion. The right half of data is expanded from 32 to 48 bits.

Next, split this key into left and right halves, $\boldsymbol{C_0}$ and $\boldsymbol{D_0}$, where each half has 28 bits.

3.7 Binary Shifting

The 48 bits are rotated left by one or 2 bits as show the following.

Binary shifting is just as its name suggests; we are **shifting** or moving **binary** values left or right. Each 1 or 0 is called a bit; which is short for **Binary** digit. BIT: The smallest unit of data in a computer. It is either a 0 or a 1. Eight bits are called a byte.

No. of cycle	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Amount of bits	1	1	2	2	2	2	2	2	1	2	2	2	2	2	1	16

3.8 Permutation Key PC-2

And the following table show from the 56 bit sub key which is output of a given round of Feistel function, only 48 bit sub key are selected.

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

Step 1: Create 16 subkeys, each of which is 48-bits long.

The 64-bit key is permuted according to the following table, **PC-1**. 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. The 4th bit of the original key is the last bit of the permuted key. Note only 56 bits of the original key appear in the permuted key.

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

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: Encode each 64-bit block of data.

There is an *initial permutation* **IP** of the 64 bits of the message data **M**. This rearranges the bits according to the following table, where the entries in the table show the new arrangement of the bits from their initial order. The 58th bit of **M** becomes the first bit of **IP**. The 50th bit of **M** becomes the second bit of **IP**. The 7th bit of **M** is the last bit of **IP**.

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

3.9 S-Blocks

1. 48 bit input is divided into 6 bit input of 8 blocks.
2. From each 6 bit the first and the last bit is taken as a row value and the remaining 4 bits are taken as a column value.
3. The resultant S- box value is a 4 bit output. For eg, for a 6 bit input 001110 the row value is 0(00) and the column value is 7(0111) and the resultant S1 box is 8 whose 4 bit output is 1000.

We have not yet finished calculating the function f . To this point we have expanded R_{n-1} from 32 bits to 48 bits, using the selection table, and XORed the result with the key K_n . We now have 48 bits, or eight groups of six bits. We now do something strange with each group of six bits: we use them as addresses in tables called "**S boxes**". Each group of six bits will give us an address in a different **S** box. Located at that address will be a 4 bit number. This 4 bit number will replace the original 6 bits. The net result is that the

eight groups of 6 bits are transformed into eight groups of 4 bits (the 4-bit outputs from the **S** boxes) for 32 bits total.
Write the previous result, which is 48 bits, in the form

Row	**Column Number**															
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**

If S_1 is the function defined in this table and B is a block of 6 bits, then $S_1(B)$ is determined as follows: The first and last bits of B represent in base 2 a number in the decimal range 0 to 3 (or binary 00 to 11). Let that number be i. The middle 4 bits of B represent in base 2 a number in the decimal range 0 to 15 (binary 0000 to 1111). Let that number be j. Look up in the table the number in the i-th row and j-th column. It is a number in the range 0 to 15 and is uniquely represented by a 4 bit block. That block is the output $S_1(B)$ of S_1 for the input B. For example, for input block B = 011011 the first bit is "0" and the last bit "1" giving 01 as the row. This is row 1. The middle four bits are "1101". This is the binary equivalent of decimal 13, so the column is column number 13. In row 1, column 13 appears 5. This determines the output; 5 is binary 0101, so that the output is 0101. Hence S_1(011011) = 0101.
The tables defining the functions $S_1,...,S_8$ are the following:

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

3.10 Permutation P

The output block of 32 bit from S -box undergo P-Permutation
A permutation is a mathematical calculation of the number of ways a particular set can be arranged, where the order of the arrangement matters.

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

3.11 Final Permutation

This is done for every block of data which is the inverse of IP.

Initial **Permutation**: A- Initial **Permutation** takes the plaintext as input. The table consists of 64 bits numbered from 1 to 64: B- Then the initial **permutation** will be permuted input as 64 bits: C- The Inverse Initial **Permutation** is: 2- The permuted input block split into two halves each is 32 bits

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

4 Conclusion

In this paper we have explained step by step and briefly mathematical procedure for DES algorithm and have also given some examples for the same. The major concern in DES algorithm security is about 2 areas such as nature of algorithm and key size. It is clear that DES can be broken using 2^{55} encryptions. However, today most applications use either 3DES with two keys or 3DES with three keys. These two multiple DES versions make DES resistant to brute- force attacks such is good for Network security.