**Available Online at www.ijcsmc.com**

**International Journal of Computer Science and Mobile Computing**

**A Monthly Journal of Computer Science and Information Technology**

**ISSN 2320–088X**
**IMPACT FACTOR: 6.199**

***IJCSMC, Vol. 8, Issue. 9, September 2019, pg.30 – 48***

# A New Approach for Data Cryptography

**Ziad Alqad[1]; Majid Oraiqat[2]; Hisham Almujafet[3]; Salah Al-Saleh[4]; Hind Al Husban[4]; Soubhi Al-Rimawi[4]**

[1]Department of Computer Engineering, Al Balqa Applied University, Amman, Jordan
[2]Department of Communications Engineering, Al Balqa Applied University, Amman, Jordan
[3]Department of Mechanical Engineering, Al Balqa Applied University, Amman, Jordan
[4]Department of Financial, Administrative and Computer Sciences, Balqa'a Applied University / Zarqa University College

***Abstract:*** ***Due to the large number of different computer applications transactions on the internet, cryptography is a vital key in ensuring the security of the transactions. Cryptography is an important way of achieving data confidentiality, data integrity, user authentication and non-repudiation.***

***In this paper we will introduce a new approach of message encryption-decryption, this approach will be implemented, and the experimental results will compared with the results of DES method of data cryptography.***

***The following features of the proposed approach will be proved:***

- ***Simplicity.***
- ***Efficiency.***
- ***High security level.***
- ***Flexibility***

***Keywords: Secret message, source color image, DES, encryption time, decryption time, security.***

## *1- Introduction*

### *1-2 DES*

Data security [1], [2] is the process of protecting data from unauthorized access and data corruption throughout its lifecycle. Data security includes data encoding-decoding, tokenization, and key management practices( key selection and key generation) that protect data across all applications, networks communications and platforms [3], [4], [5].

Symmetric methods of data protection acts as shown in figure 1 by selecting a private secret key to be used by both the sender and receiver [6], [7], these methods basically applied the same procedures such as [8], [9] :

 

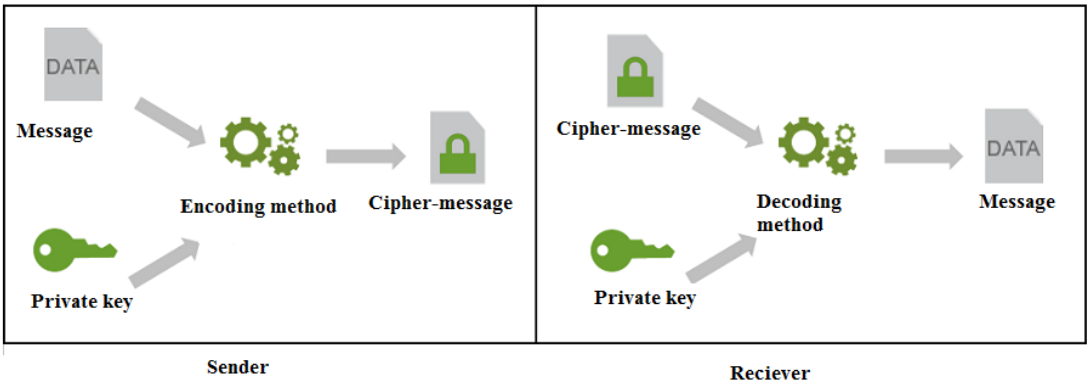


Figure 1: Symmetric data encoding-decoding

- Dividing the data message into blocks with 64 bits long.
- Selecting a private key with fixed length (64 bits for DES method).
- Key generation to define subkeys.
- Initial permutation of blocks.
- Breakdown of the blocks into two parts left and right.
- Permutation and substitution **steps** repeated 16 rounds, applying logical and mathematical operations in each round using a defined feistel function and s-box.
- Re-joining of the left and right parts to form the decoded block of the message.
- Applying inverse procedures to get the encoded block.

Data encryption standard (DES) is one of the simplest methods of message encoding-decoding (encryption-decryption) [10] , [11], it uses (as shown in figure 2) a data block of 64 bits long with a private key of 56 bits long The block of the message is divided into two halves. The right half is expanded from 32 to 48 bits using another fixed table. The result is combined with the subkey for that round using the XOR operation. Using the S-boxes the 48 resulting bits are then transformed again to 32 bits, which are subsequently permutated again using yet another fixed table. This by now thoroughly shuffled right half is now combined with the left half using the XOR operation. In the next round, this combination is used as the new left half. Figure 2shows how this method works.

 

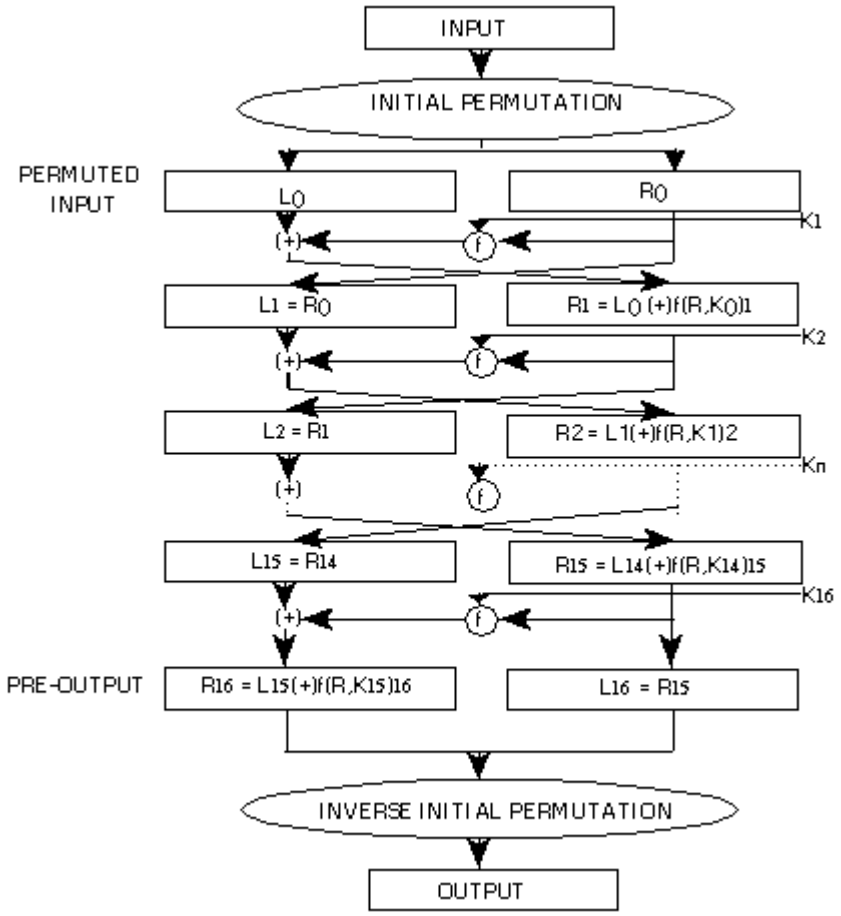


Figure 2: DES operations

To explain DES operations we will introduce a practical example.

***Worked example:***

Suppose we have a plaintext M which is equal: 85E813540F0AB405

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

First we divide this block into half left and half right:

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

Then we select a 64 bits key K:

**K** = 00010011 00110100 01010111 01111001 10011011 10111100 11011111 11110001

 

**PC-1**

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

Since the first entry in the table is "57", this means that the 57th bit of the original key **K** becomes the first bit of the permuted key **K**+. The 49th bit of the original key becomes the second bit of the permuted key. 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.

**K** = 00010011 00110100 01010111 01111001 10011011 10111100 11011111 11110001

We get a 56 bits key as:

**K**+ = 1111000 0110011 0010101 0101111 0101010 1011001 1001111 0001111

We split this key into 2 halves, where each half is 28 bits long:

$C_0$ = 1111000 0110011 0010101 0101111
$D_0$ = 0101010 1011001 1001111 0001111

With $C_0$ and $D_0$ selected, we can create sixteen blocks $C_n$ and $D_n$, 1<=$n$<=16. Each pair of blocks $C_n$ and $D_n$ is formed from the previous pair $C_{n-1}$ and $D_{n-1}$, respectively, for $n$ = 1, 2... 16, using the following schedule of "left shifts" of the previous block. To do a left shift, move each bit one place to the left, except for the first bit, which is cycled to the end of the block.

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

This means, for example, $C_3$ and $D_3$ are obtained from $C_2$ and $D_2$, respectively, by two left shifts, and $C_{16}$ and $D_{16}$ are obtained from $C_{15}$ and $D_{15}$, respectively, by one left shift. In all cases, by a single left shift is meant a rotation of the bits one place to the left, so that after one left shift the bits in the 28 positions are the bits that were previously in positions 2, 3,..., 28, 1.

 

$C_0$ = 11110000110011001010101111
$D_0$ = 01010101011001100111100011111

$C_1$ = 11100001100110010101011111
$D_1$ = 10101010110011001111000111110

$C_2$ = 11000011001100101010111111
$D_2$ = 01010101100110011111000111101

$C_3$ = 00001100110010101010111111111
$D_3$ = 01010110011001111000111110101

$C_4$ = 00110011001010101011111111100
$D_4$ = 01011001100111100011111010101

$C_5$ = 11001100101010101111111110000
$D_5$ = 01100110011110001111010101010

$C_6$ = 00110010101010111111111000011
$D_6$ = 10011001111000111110101010101

$C_7$ = 11001010101011111111100001100
$D_7$ = 01100111100011111010101010110

$C_8$ = 00101010101111111110000110011
$D_8$ = 10011110001111010101010110011

$C_9$ = 01010101011111111100001100110
$D_9$ = 00111100011110101010110011

$C_{10}$ = 01010101111111110000110011001
$D_{10}$ = 11110001111010101010110011001100

$C_{11}$ = 01010111111110000110011001010
$D_{11}$ = 11000111101010101011001100111

 

$C_{12}$ = 010111111110000110011001010 1

$D_{12}$ = 000111101010101011001100 1111

$C_{13}$ = 011111111000011001100101010 1

$D_{13}$ = 011110101010101100110011110 0

$C_{14}$ = 111111100001100110010101010 1

$D_{14}$ = 111010101010110011001111000 1

$C_{15}$ = 111110000110011001010101011 1

$D_{15}$ = 101010101011001100111100011 1

$C_{16}$ = 111100001100110010101010111 1

$D_{16}$ = 010101010110011001111000111 1

We now form the keys $K_n$, for 1<=$n$<=16, by applying the following permutation table to each of the concatenated pairs $C_nD_n$. Each pair has 56 bits, but **PC-2** only uses 48 of these.

**PC-2**

| | | | | | |
|---|---|---|---|---|---|
| 14 | 17 | 11 | 24 | 1 | 5 |
| 3 | 28 | 15 | 6 | 21 | 10 |
| 23 | 19 | 12 | 4 | 26 | 8 |
| 16 | 7 | 27 | 20 | 13 | 2 |
| 41 | 52 | 31 | 37 | 47 | 55 |
| 30 | 40 | 51 | 45 | 33 | 48 |
| 44 | 49 | 39 | 56 | 34 | 53 |
| 46 | 42 | 50 | 36 | 29 | 32 |

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 $C_nD_n$.

For the first key we have

$C_1D_1$ = 1110000 1100110 0101010 1011111 1010101 0110011 0011110 0011110

Which, after we apply the permutation **PC-2**, becomes

$K_1$ = 000110 110000 001011 101111 111111 000111 000001 110010

For the other keys we have

$K_2$ = 011110 011010 111011 011001 110110 111100 100111 100101
$K_3$ = 010101 011111 110010 001010 010000 101100 111110 011001
$K_4$ = 011100 101010 110111 010110 110110 110011 010100 011101
$K_5$ = 011111 001110 110000 000111 111010 110101 001110 101000
$K_6$ = 011000 111010 010100 111110 010100 000111 101100 101111
$K_7$ = 111011 001000 010010 110111 111101 100001 100010 111100
$K_8$ = 111101 111000 101000 111010 110000 010011 101111 111011
$K_9$ = 111000 001101 101111 101011 111011 011110 011110 000001
$K_{10}$ = 101100 011111 001101 000111 101110 100100 011001 001111
$K_{11}$ = 001000 010101 111111 010011 110111 101101 001110 000110
$K_{12}$ = 011101 010111 000111 110101 100101 000110 011111 101001
$K_{13}$ = 100101 111100 010111 010001 111110 101011 101001 000001
$K_{14}$ = 010111 110100 001110 110111 111100 101110 011100 111010
$K_{15}$ = 101111 111001 000110 001101 001111 010011 111100 001010
$K_{16}$ = 110010 110011 110110 001011 000011 100001 011111 110101

 

**Now we 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 |

Applying the initial permutation to the block of text **M**, given previously, we get

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

Here the 58th bit of **M** is "1", which becomes the first bit of **IP**. The 50th bit of **M** is "1", which becomes the second bit of **IP**. The 7th bit of **M** is "0", which becomes the last bit of **IP**.

Next divide the permuted block **IP** into a left half $L_0$ of 32 bits, and a right half $R_0$ of 32 bits.

From **IP**, we get $L_0$ and $R_0$

$L_0$ = 1100 1100 0000 0000 1100 1100 1111 1111
$R_0$ = 1111 0000 1010 1010 1111 0000 1010 1010

We now proceed through 16 iterations, for 1<=$n$<=16, using a function $f$ which operates on two blocks--a data block of 32 bits and a key $K_n$ of 48 bits--to produce a block of 32 bits. **Let + denote XOR addition, (bit-by-bit addition modulo 2)**. Then for **n** going from 1 to 16 we calculate

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

This results in a final block, for $n$ = 16, of $L_{16}R_{16}$. In each iteration we take the right 32 bits of the previous result and make them the left 32 bits of the current step. For the right 32 bits in the current step, we XOR the left 32 bits of the previous step with the calculation $f$.

For $n$ = 1, we have

$K_1$ = 000110 110000 001011 101111 111111 000111 000001 110010
$L_1 = R_0$ = 1111 0000 1010 1010 1111 0000 1010 1010
$$R_1 = L_0 + f(R_0, K_1)$$

It remains to explain how the function $f$ works. To calculate $f$, we first expand each block $R_{n-1}$ from 32 bits to 48 bits. This is done by using a selection table that repeats some of the bits in $R_{n-1}$. We'll call the use of this selection table the function **E**. Thus **E** ($R_{n-1}$) has a 32 bit input block, and a 48 bit output block.

 

Let **E** be such that the 48 bits of its output, written as 8 blocks of 6 bits each, are obtained by selecting the bits in its inputs in order according to the following table:

**E BIT-SELECTION TABLE**

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

Thus the first three bits of **E** ($\boldsymbol{R_{n-1}}$) are the bits in positions 32, 1 and 2 of $\boldsymbol{R_{n-1}}$ while the last 2 bits of **E** ($\boldsymbol{R_{n-1}}$) are the bits in positions 32 and 1.

We calculate **E** ($\boldsymbol{R_0}$) from $\boldsymbol{R_0}$ as follows:

$$\boldsymbol{R_0} = 1111\ 0000\ 1010\ 1010\ 1111\ 0000\ 1010\ 1010$$
$$\mathbf{E}(\boldsymbol{R_0}) = 011110\ 100001\ 010101\ 010101\ 011110\ 100001\ 010101\ 010101$$

(Note that each block of 4 original bits has been expanded to a block of 6 output bits.)

Next in the ***f*** calculation, we XOR the output **E** ($\boldsymbol{R_{n-1}}$) with the key $\boldsymbol{K_n}$:

$$\boldsymbol{K_n} + \mathbf{E}(\boldsymbol{R_{n-1}}).$$

For $\boldsymbol{K_1}$ , **E** ($\boldsymbol{R_0}$), we have

$$\boldsymbol{K_1} = 000110\ 110000\ 001011\ 101111\ 111111\ 000111\ 000001\ 110010$$
$$\mathbf{E}(\boldsymbol{R_0}) = 011110\ 100001\ 010101\ 010101\ 011110\ 100001\ 010101\ 010101$$
$$\boldsymbol{K_1}+\mathbf{E}(\boldsymbol{R_0}) = 011000\ 010001\ 011110\ 111010\ 100001\ 100110\ 010100\ 100111$$

We have not yet finished calculating the function ***f*** . To this point we have expanded $\boldsymbol{R_{n-1}}$ from 32 bits to 48 bits, using the selection table, and XORed the result with the key $\boldsymbol{K_n}$ . 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:

$$\boldsymbol{K_n} + \mathbf{E}(\boldsymbol{R_{n-1}}) = \boldsymbol{B_1B_2B_3B_4B_5B_6B_7B_8},$$

Where each $\boldsymbol{B_i}$ is a group of six bits. We now calculate

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

where $\boldsymbol{S_i(B_i)}$ referrers to the output of the ***i***-th **S** box.

 

To repeat, each of the functions ***S1, S2... S8***, takes a 6-bit block as input and yields a 4-bit block as output. The table to determine $S_1$ is shown and explained below:

**S1**

| Row No. | Column Number 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 |

For the first round, we obtain as the output of the eight **S** boxes:

$K_1 + \mathbf{E}(R_0) = 011000\ 010001\ 011110\ 111010\ 100001\ 100110\ 010100\ 100111.$

$S_1(B_1)S_2(B_2)S_3(B_3)S_4(B_4)S_5(B_5)S_6(B_6)S_7(B_7)S_8(B_8) = 0101\ 1100\ 1000\ 0010\ 1011\ 0101\ 1001\ 0111$

The final stage in the calculation of $f$ is to do a permutation **P** of the **S**-box output to obtain the final value of $f$:

$$f = \mathbf{P}(S_1(B_1)S_2(B_2)...S_8(B_8))$$

The permutation **P** is defined in the following table. **P** yields a 32-bit output from a 32-bit input by permuting the bits of the input block.

**P**

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

 

From the output of the eight **S** boxes:

$S_1(B_1)S_2(B_2)S_3(B_3)S_4(B_4)S_5(B_5)S_6(B_6)S_7(B_7)S_8(B_8)$ = 0101 1100 1000 0010 1011 0101 1001 0111

We get

$f$ = 0010 0011 0100 1010 1010 1001 1011 1011

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

= 1100 1100 0000 0000 1100 1100 1111 1111
+ 0010 0011 0100 1010 1010 1001 1011 1011
= 1110 1111 0100 1010 0110 0101 0100 0100

In the next round, we will have $L_2 = R_1$, which is the block we just calculated, and then we must calculate $R_2 = L_1 + f(R_1, K_2)$, and so on for 16 rounds. At the end of the sixteenth round we have the blocks $L_{16}$ and $R_{16}$. We then ***reverse*** the order of the two blocks into the 64-bit block

$$R_{16}L_{16}$$

And apply a final permutation $\mathbf{IP}^{-1}$ as defined by the following table:

$\mathbf{IP}^{-1}$

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

That is, the output of the algorithm has bit 40 of the pre-output block as its first bit, bit 8 as its second bit, and so on, until bit 25 of the pre-output block is the last bit of the output.

If we process all 16 blocks using the method defined previously, we get, on the 16th round,

$L_{16}$ = 0100 0011 0100 0010 0011 0010 0011 0100
$R_{16}$ = 0000 1010 0100 1100 1101 1001 1001 0101

We reverse the order of these two blocks and apply the final permutation to

$R_{16}L_{16}$ = 00001010 01001100 11011001 10010101 01000011 01000010 00110010 00110100

$IP^{-1}$ = 10000101 11101000 00010011 01010100 00001111 00001010 10110100 00000101

Which is in hexadecimal format:

85E813540F0AB405.

 

This is the encrypted form of **M** = 0123456789ABCDEF: namely, **C** = 85E813540F0AB405.

Decryption is simply the inverse of encryption, following the same steps as above, but reversing the order in which the subkeys are applied.

### *1-2 Digital color images*

Digital color image is an important type of data used over the internet, every color image is consisted of three channels [12], [13], the first channel is the red color, the second is the green color, while the third one is the blue color, mixing these colors to gather forms the pixel color [14], [15].

Digital color image in processing phase can be considered as 3D matrix, the first dimension is reserved for the red color, the second dimension is reserved for the green color, and the third one is reserved for the blue color[16], [17].

Each color image can be studied using its histogram which is an array that contains the repetitions of each color value (0 to 255), one histogram can calculated for each color [16], [17] as shown in figure 3.

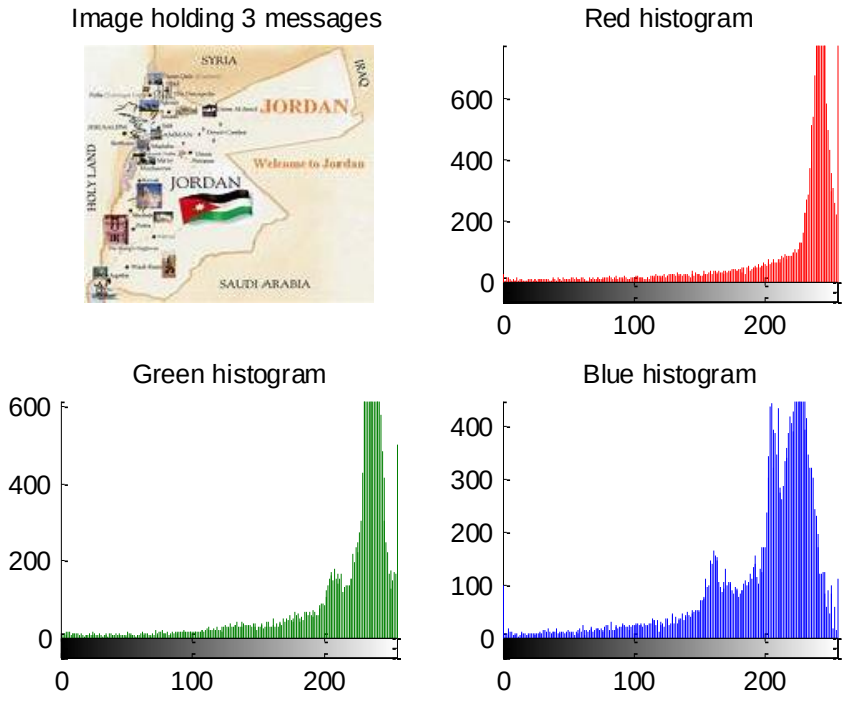


Figure 3: Color image and histograms

If the color image is equalized [18] then the colors values are normally distributed, and the histogram will contains all the values within the range 0 to 255 as shown in figure 4, thus the histogram of the color image can cover all the characters in ASCII table, which means that the color image can be considered as a bank of data which can contain any message with any combination of letters, this fact will be considered in the proposed novel approach of message encryption-decryption, and here instead of using a private key we can use a color image as a source which supply us with the needed codes for any given secret massage as shown in figure 5.

 

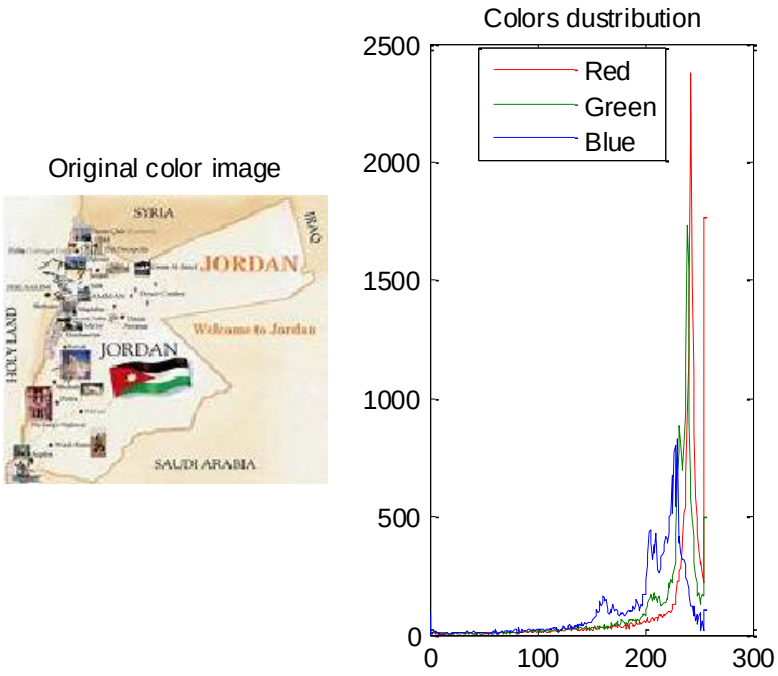


Figure 4: Colors distribution.

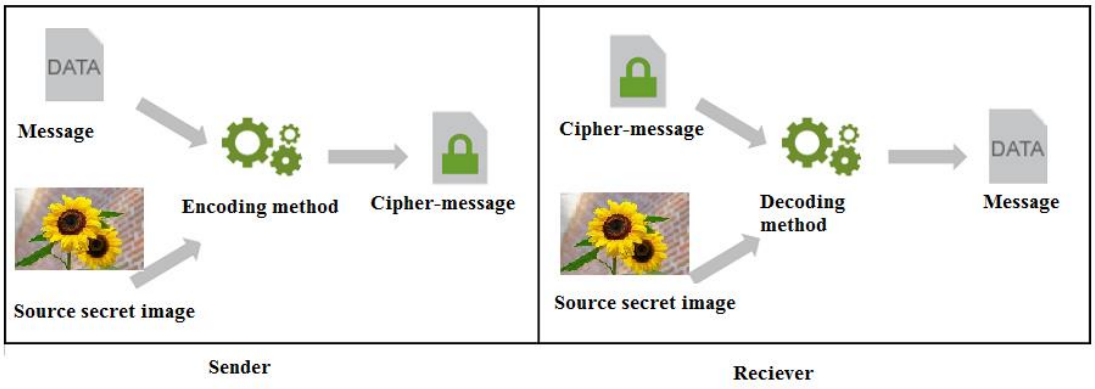


Figure 5: New approach of message encryption-decryption.

### *2- The Proposed Approach*

Based on the normal color distribution in the image histogram we can use the pixel position (row, column and color channel number) to decode any character from ASCII table, these codes will be fixed for the selected source image and the selected secret message , to apply the coding we have to perform the following steps:

1) Select the source secret color image
2) Equalize the image if necessary.
3) Get the secret message.
4) Initialize the code matrix.
5) For each character in the secret message do the following:
    a) Get the decimal value of the character.
    b) Find the first appearance of the decimal value in the image.
    c) Retrieve the position values.

 

d) Add the retrieved position values to the code matrix.

6) Save the code matrix (the encrypted message).

The decryption phase can be implemented applying the following steps:

1) Get the source secret image (must be the same image used in the encryption phase).
2) Get the encrypted message.
3) Initialize the decrypted massage.
4) For each row in the encrypted message do the following.

   a) Get the row, column and color channel.
   b) Use the retrieved position to get the character value from the image.
   c) Concatenate the obtained value to decrypted message.

5) Change the decimal values of the obtained message to characters to get the original message.

Using this approach we have to pay attention on the following facts:

- One color image can be used as source image to get the encrypted form of any message with any combination of characters, as shown in figure 6:

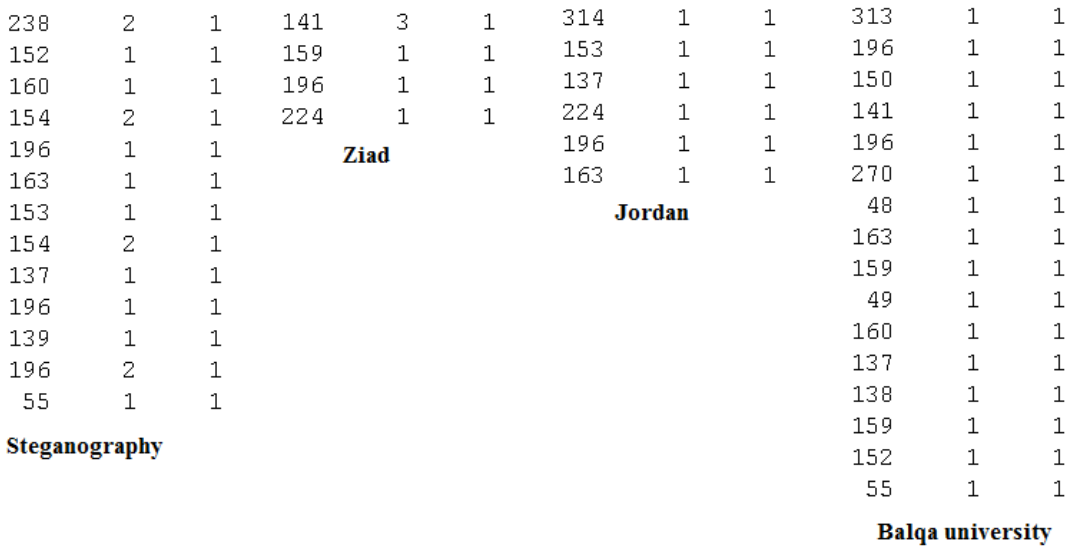


Figure 6: Messages encryption using the same source image.

- Every image encodes the message depending on its own data, so various images generate various codes for the same character as shown in table 1.
- One image can be used to encode a message with unlimited size (the message size may be even bigger than the image size).
- The correct encrypted message can be obtained only by using the desired source image. This image is only known by the sender and receiver.
- The proposed approach provides a high level of security, because it is very difficult to guess the source image. Here the number of combination used for guessing equal $256^{r+c+3}$ ( r:number of rows, c:number of columns, 3 : number of color in the source image), if the image with size 400x300x3 then the number of combinations is a very huge number and it is equal to:

9.8332167945635864126735468519358e+1692 (very big number!!).

- The proposed approach is very simple, and it is very easy to implement it.

 

- The proposed approach is very efficient and will show this in the implementation part.

Table 1: Different codes using different images

| Character | Decimal valu | Decrypted Character using image 6 | | | Decrypted Character using image 1 | | |
|---|---|---|---|---|---|---|---|
| | | **Row** | **Column** | **Color** | **Row** | **Column** | **Color** |
| A | 65 | **308** | **1** | **1** | **133** | **14** | **1** |
| B | 66 | **313** | **1** | **1** | **102** | **18** | **1** |
| C | 67 | **205** | **3** | **1** | **133** | **7** | **1** |
| D | 68 | **226** | **3** | **1** | **63** | **6** | **1** |
| E | 68 | **306** | **1** | **1** | **132** | **14** | **1** |
| F | 70 | **300** | **1** | **1** | **92** | **6** | **1** |
| G | 71 | **197** | **2** | **1** | **61** | **35** | **1** |
| H | 72 | **231** | **2** | **1** | **132** | **13** | **1** |
| I | 73 | **220** | **2** | **1** | **63** | **7** | **1** |
| J | 74 | **314** | **1** | **1** | **136** | **8** | **1** |
| K | 75 | **197** | **1** | **1** | **136** | **7** | **1** |
| L | 76 | **207** | **2** | **1** | **89** | **35** | **1** |
| M | 77 | **228** | **2** | **1** | **72** | **4** | **1** |
| N | 78 | **228** | **3** | **1** | **132** | **8** | **1** |
| O | 79 | **227** | **1** | **1** | **87** | **3** | **1** |
| P | 80 | **213** | **2** | **1** | **101** | **23** | **1** |
| Q | 81 | **240** | **1** | **1** | **145** | **11** | **1** |
| R | 82 | **213** | **1** | **1** | **72** | **3** | **1** |
| S | 83 | **238** | **2** | **1** | **97** | **5** | **1** |
| T | 84 | **228** | **1** | **1** | **97** | **3** | **1** |
| U | 85 | **220** | **1** | **1** | **148** | **8** | **1** |
| V | 86 | **222** | **3** | **1** | **142** | **14** | **1** |
| W | 87 | **198** | **1** | **1** | **133** | **13** | **1** |
| X | 88 | **225** | **1** | **1** | **94** | **3** | **1** |
| Y | 89 | **210** | **1** | **1** | **77** | **4** | **1** |
| Z | 90 | **141** | **3** | **1** | **60** | **5** | **1** |

### *3- Implementation and Experimental Results*

For comparisons purposes we select DES method of data encryption-decryption, because it is one of the simplest method among the other used symmetric methods of data cryptography.

DES method was implemented using matlab, figure 7 shows the GUI of the program, while table 2 shows the efficiency parameters measured for this method using various messages with a private key=931955.

Table 2: Efficiency parameters for DES method

| Message length | Encryption time(seconds) | Decryption time(seconds) |
|---|---|---|
| 11 | 0.1480 | 0.0600 |
| 26 | 0.1800 | 0.1000 |
| 35 | 0.2190 | 0.1190 |
| 48 | 0.2390 | 0.1530 |
| 60 | 0.3200 | 0.1680 |
| 64 | 0.3210 | 0.1690 |
| 72 | 0.3670 | 0.1860 |
| 80 | 0.4650 | 0.1870 |
| 85 | 0.5190 | 0.2040 |
| 90 | 0.5930 | 0.2090 |
| 95 | 0.6170 | 0.2127 |
| 100 | 0.6540 | 0.2290 |
| **Average :63.8333** | **0.3868** | **0.1664** |

 

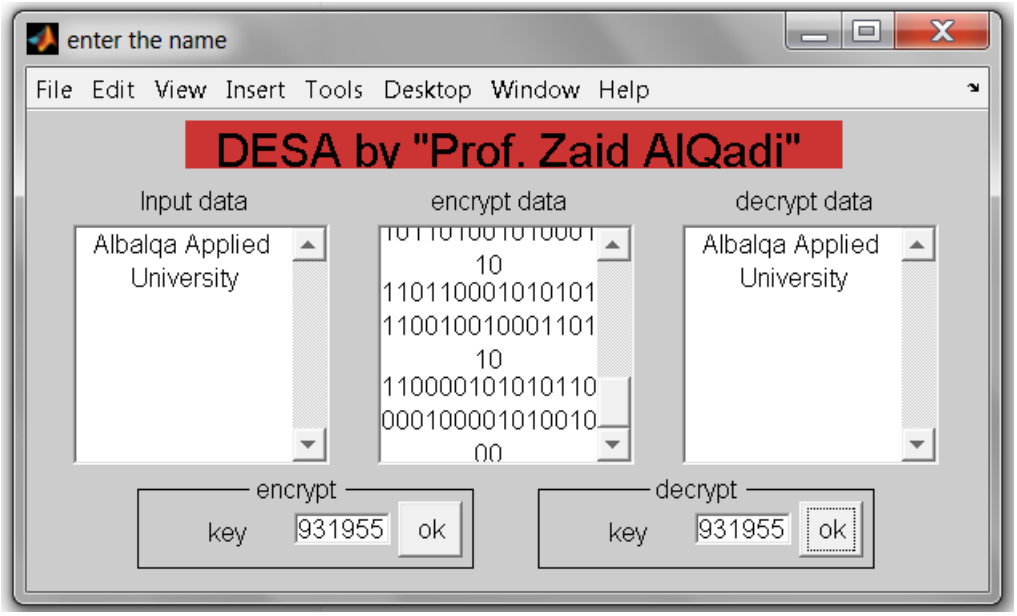


Figure 7: GUI of DES method

Here we have to notice that the private key long does not affect the measures times.

Here we can see that both the encryption and decryption times grow with message length growing as shown in figure 8.

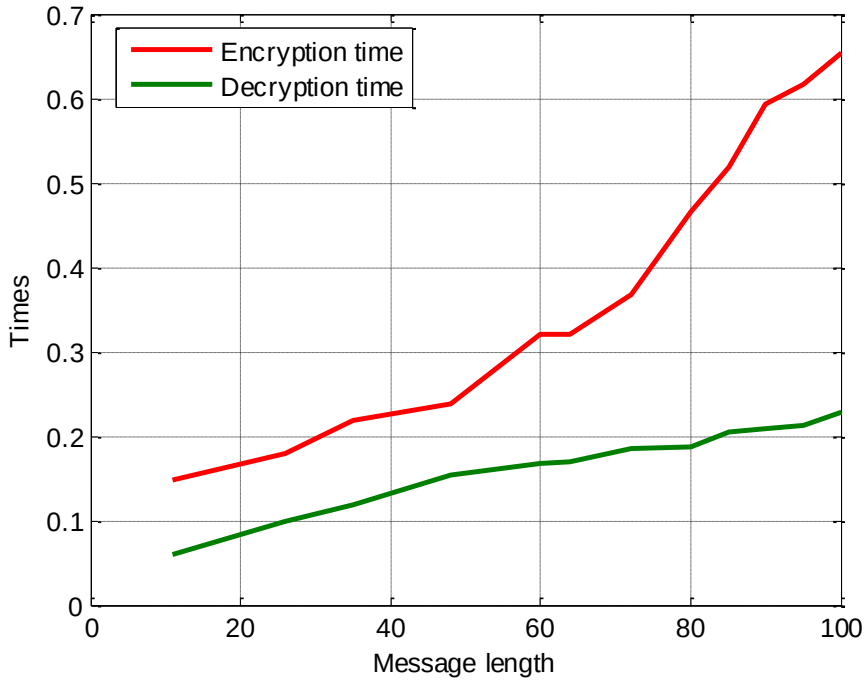


Figure 8: Encryption-decryption times

The new approach was also implemented using matlab; figure 9 shows the GUI of this implementation

 


Figure 9: New approach GUI

The image shown in figure 10 was selected as a source image and the same experiment used in DES was repeated using the new approach, table 3 show the experimental results of the implementation.

F

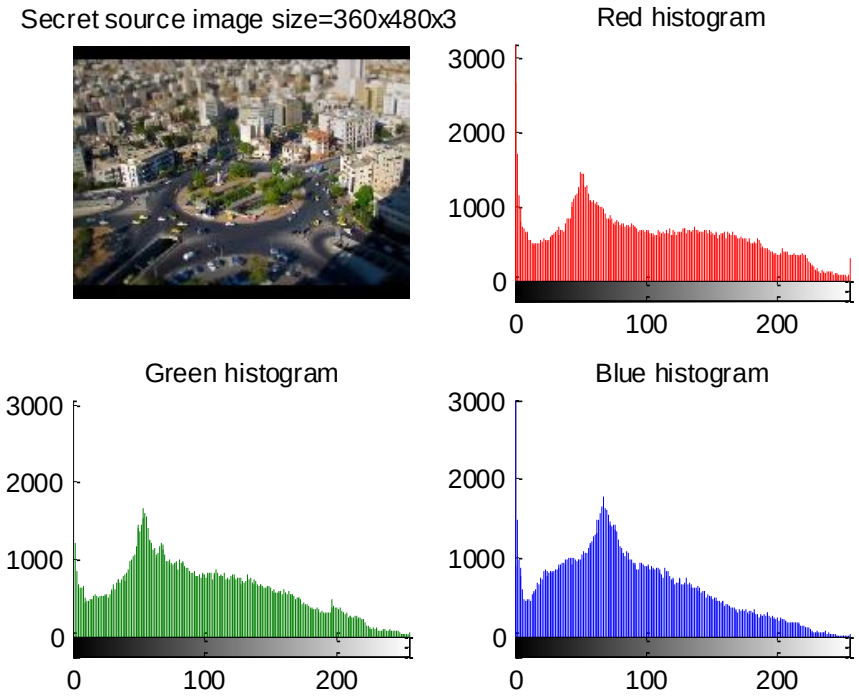


Figure 10: Source color image

 

Table 3: Efficiency parameters for the new approach

| Message length | Encryption time(seconds) | Decryption time(seconds) |
|---|---|---|
| 11 | 0.008000 | 0.000001 |
| 26 | 0.016000 | 0.000001 |
| 35 | 0.020000 | 0.001000 |
| 48 | 0.027000 | 0.001300 |
| 60 | 0.032000 | 0.001400 |
| 64 | 0.035000 | 0.001400 |
| 72 | 0.037000 | 0.001500 |
| 80 | 0.041000 | 0.001700 |
| 85 | 0.046000 | 0.001750 |
| 90 | 0.048000 | 0.001800 |
| 95 | 0.061000 | 0.001830 |
| 100 | 0.068000 | 0.001880 |
| **Average :63.8333** | **0.0366** | **0.0013** |
| **Speed up of the new approach** | **0.3868/0.0366=10.5683 times** | **0.1664/0.0013=128 times** |

Figure 11 illustrates the relationship between encryption, decryption times and message length.

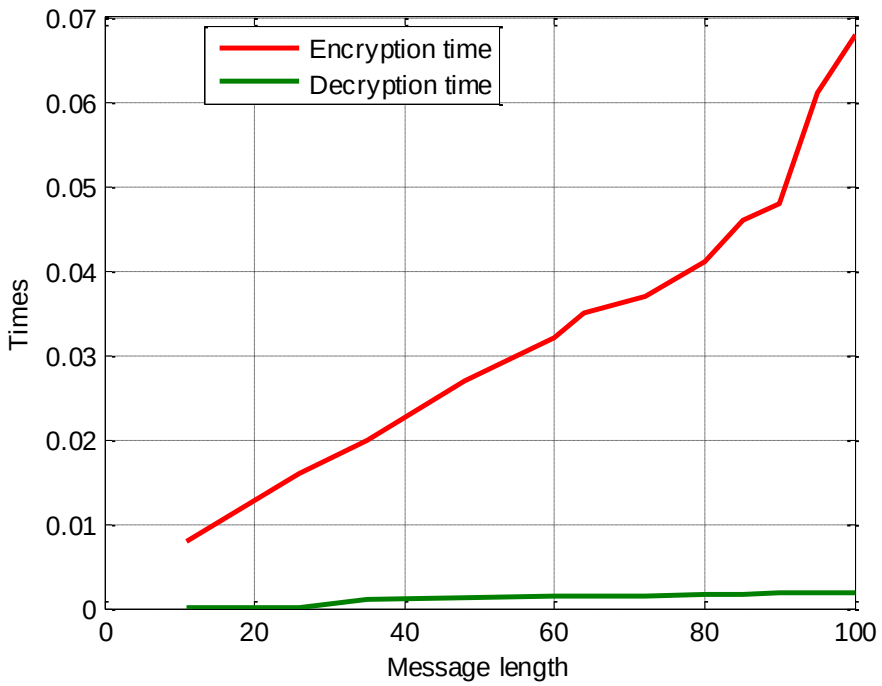


Figure 11: Relationship between encryption, decryption times and message length

From the obtained results we can see that in average the proposed approach is faster in **10.5683** times in encryption phase comparing with DES method, and faster in **128** times in decryption phase comparing with DES method

***Conclusions***

A new approach of secret messages encryption-decryption was proposed, tested and implemented; the obtained experimental results showed and proved the following facts:

- The proposed approach can be considered as a symmetric method of data cryptography.
- The same source color image can be used to encrypt-decrypt various messages.

 

- The message size to be encrypted is unlimited and it may exceed the image size.
- The encryption process does not depend on the message contents.
- The security level is very high, because it is very difficult to guess the source image.
- The propose approach is very simple and does not require any arithmetic and logical operations, it also does not require any extra data structure such S-boxes.
- The propose approach provides a high efficiency comparing with DES method.

# References

[1] Ziad A. Alqadi, Majed O. Al-Dwairi, Amjad A. Abu Jazar and Rushdi Abu Zneit, Optimized True-RGB color Image Processing, World Applied Sciences Journal 8 (10): 1175-1182, ISSN 1818-4952, 2010.
[2] A. A. Moustafa, Z. A. Alqadi, Color Image Reconstruction Using A New R'G'I Model, journal of Computer Science, Vol.5, No. 4, pp. 250-254, 2009.
[3] Jamil Al Azzeh, Hussein Alhatamleh, Ziad A. Alqadi, Mohammad Khalil Abuzalata, Creating a Color Map to be used to Convert a Gray Image to Color Image; International Journal of Computer Applications , November 2016,Volume 153,Issue 2.
[4] Jamil Al-Azzeh, Ziad Alqadi, Mohammed Abuzalata, Performance Analysis of Artificial Neural Networks used for Color Image Recognition and Retrieving, International Journal of Computer Science and Mobile Computing,2019,Volume 8 Issue 2.
[5] Jamil AL-Azzeh, Bilal Zahran, Ziad Alqadi, Belal Ayyoub and Mazen Abu-Zaher: A Novel Zero-Error Method to Create a Secret Tag for an Image; Journal of Theoretical and Applied Information Technology 15th July 2018.
[6] Jamil AL-Azzeh, Bilal Zahran and Ziad Alqadi: Salt and Pepper Noise: Effects and Removal, International Journal on Informatics Visualization July 2018, Volume 2 Issue 4.
[7] Musbah J. Aqel , Ziad A. Alqadi, Ibraheim M. El Emary ,Analysis of Stream Cipher Security Algorithm, Journal of Information and Computing Science Vol. 2, No. 4, 2007, pp. 288-298.
[8] Belal Ayyoub, Ashraf Abu-Ein, Ziad Alqadi, Suggested Method to Create Color Image Features Victor, Journal of Engineering and Applied Sciences, 2019, Volume14, Issue7.
[9] K Matrouk, A Al-Hasanat, H Alasha'ary, Z. Al-Qadi,H Al-Shalabi, Speech fingerprint to identify isolated word person, World Applied Sciences Journal, Vol. 31, No. 10, pp. 1767-1771, 2014.
[10] Mohammed Abuzalata, Ziad Alqadi; Jamil Al-Azzeh; Qazem Jaber, Modified Inverse LSB Method for Highly Secure Message Hiding, *IJCSMC, Vol. 8, Issue. 2, February 2019, pg.93 – 103*
[11] Mutaz Rasmi Abu Sara Rashad J. Rasras, Ziad A. AlQadi, Engineering, A Methodology Based on Steganography and Cryptography to Protect Highly Secure Messages Technology & Applied Science Research, Vol.9 Issue 1, Pages 3681-3684, 2019.
[12] Ziad Alqadi, Bilal Zahran, Qazem Jaber, Belal Ayyoub, Jamil Al-Azzeh, Ahmad Sharadqh, proposed Implementation Method to Improve LSB Efficiency, International Journal of Computer Science and Mobile Computing, Vol.8 Issue.3, March-2019, pg. 306-319.
[13] Deepak Garg, Gourav Sharma, Applications of Steganography in Information Hiding, international Journal of Advanced Research in Education & Technology (IJARET) 12 Vol. 3, Issue 1 (Jan. - Mar. 2016).
[14] J. Al-Azzeh, B. Zahran, Z. Alqadi, B. Ayyoub, M. Abu-Zaher, "A Novel zero-error method to create a secret tag for an image", Journal of Theoretical and Applied Information Technology, Vol . 96. No. 13, pp. 4081-4091, 2018.
[15] Prof. Ziad A.A. Alqadi, Prof. Mohammed K. Abu Zalata, Ghazi M. Qaryouti, Comparative Analysis of Color Image Steganography, JCSMC, Vol.5, Issue. 11, November 2016, pg.37–43.
[16] M. Jose, "Hiding Image in Image Using LSB Insertion Method with Improved Security and Quality", International Journal of Science and Research, Vol. 3, No. 9, pp. 2281-2284, 2014.
[17] Ziad Alqadi; Bilal Zahran; Qazem Jaber; Belal Ayyoub; Jamil Al-Azzeh, Enhancing the Capacity of LSB Method by Introducing LSB2Z Method; International Journal of Computer Science and Mobile Computing,2019,Volume 8 Issue 3.
[18] Naseem Asad, Ismail Shayeb, "A Modification of Least Significant Digit (LSD) Digital Watermark Technique", International Journal of Computer Applications
Volume 179 No.32, April 2018
[19] Jamil Al Azzeh, Ziad Alqadi Qazem, M Jabber;"Statistical Analysis of Methods Used to Enhanced Color Image"; XX International Scientific and Technical Conference;2016.
[20] Mazen Abuzaher Jamil Al-Azzeh, :JPEG Based Compression Algorithm"; International Journal of Engineering and Applied Sciences, 2017,Volume 4 Issue 4.
[21] Jamil Al-Azzeh, Ziad Alqadi, Qazem Jaber; "A Simple, Accurate and Highly Secure Method to Encrypt-Decrypt Digital Images"; JOIV: International Journal on Informatics Visualization 2019.

 