Chapter

Hybrid Approaches to Block Cipher

Roshan Chitrakar, Roshan Bhusal and Prajwol Maharjan

Abstract

This chapter introduces two new approaches to block cipher—one is DNA hybridization encryption scheme (DHES) and the other is hybrid graphical encryption algorithm (HGEA). DNA cryptography deals with the techniques of hiding messages in the form of a DNA sequence. The key size of data encryption standard (DES) can be increased by using DHES. In DHES, DNA cryptography algorithm is used for encryption and decryption, and one-time pad (OTP) scheme is used for key generation. The output of DES algorithm is passed as an input to DNA hybridization scheme to provide an added security. The second approach, HGEA, is based on graphical pattern recognition. By performing multiple transformations, shifting and logical operations, a block cipher is obtained. This algorithm is influenced by hybrid cubes encryption algorithm (HiSea). Features like graphical interpretation and computation of selected quadrant value are the unique features of HGEA. Moreover, multiple key generation scheme combined with graphical interpretation method provides an increased level of security.

Keywords: DNA hybridization encryption scheme (DHES), hybrid graphical encryption algorithm (HGEA), DNA cryptography, data encryption standard (DES), one-time pad (OTP), hybrid cube encryption algorithm (HiSea), block cipher

1. Introduction

There exist a number of cryptographic techniques for secure data communication [1], but many are vulnerable to attacks. With the failure of cryptographic algorithms like data encryption standard (DES), new approaches to cipher security are needed [2, 3]. A cryptographic scheme can be made more secure by combining it with relatively secure techniques. Theoretically, this hybridization method can be applied to any cryptographic scheme but block ciphers provide more rounds for working in terms of permutation and combination.

DNA-based method [4, 5] is one such approach that along with one-time pad (OTP) scheme can be applied to DES. OTP is the only unbreakable encryption that uses polyalphabetic randomness for the key [6]. So, OTP can be combined with DNA cryptography by taking longer message and key size ($\geq$ 64 bit) so as to make brute force attack difficult and impractical [7].

As the first part of this chapter, DNA hybridization encryption scheme (DHES) is described, in which an improved algorithm named DDHO (that stands for DES and DNA-based hybridization with OTP) is proposed.

Another technique that can be combined with DES is “hybrid graphical encryption algorithms (HGEA), which is based on graphical interpretation by pattern recognition and transformation like hybrid cubes encryption algorithm (HiSea) [8, 9]. Most of the graphical encryption algorithms use mono-alphabetic or poly-alphabetic substitution and their range of input values is limited. But, HGEA uses a range of characters consisting of 256 possible values. It also produces output of 256 characters for single-input plaintext. Moreover, HGEA can be used by software as well as realized by implementing hardware devices.

The rest of this chapter is organized as follows. Section 2 describes DNA hybridization encryption scheme in detail. This section also presents performance analysis of algorithms and methods used by DHES. Similarly, Section 3 presents hybrid graphical encryption with illustrations. This section also presents performance analysis of encryption and decryption algorithms used by HGEA by comparing it with that of DES.

2. DNA hybridization encryption scheme (DHES)

2.1 DNA cryptography

DNA cryptography is an emerging field of cryptography that deals with hiding data in terms of DNA sequences [10, 11]. It can be implemented by using modern biological techniques as tools and DNA as information carrier to fully exert the inherent advantage of high storage density and high parallelism to achieve encryption [12]. The DNA cryptography uses the concept of molecular approach in traditional cryptographic technique to make the system more secure.

Some terminologies used in biochemical operations of DNA cryptography are annealing, melting, ligation, amplification, cutting, gel electrophoresis, oligonucleotides, etc. [13, 14].

2.2 DNA hybridization

DNA hybridization is a process in which two single-stranded DNAs (ssDNAs) are combined together to produce a single DNA sequence [15]. The two ssDNAs are complementary to each other and are of same length. That means: if one strand of DNA is $3'$ to $5'$, then the other strand must be $5''$. If not, then the hybridization of the pairs fails and fragmentation occurs. To remove such fragmentation, fragment assembly has to be done [16].

2.2.1 OTP scheme

OTP is the only potentially unbreakable encryption method. Plaintext encrypted using an OTP cannot be retrieved without the encryption key. The key generated by an OTP must be random and generated by a non-deterministic, non-repeatable process. The key also must be never re-used. In OTP scheme, the length of the key must be greater than or equal to the length of the plaintext.

2.2.2 Key generation

OTP is used for key generation. The generated key is unique and only the sender and receiver know the key and that generated key is destroyed once it is used. The key generated by computers is not truly random; so, a pseudorandom number generator function is used for generating the key and the generated key is in the DNA. The size of ssDNA key is greater than the original size of the message, which results in a longer size of the encrypted message and that makes it difficult to break. The length of the key is the result of multiplication of the number of bits required to represent each character and total number of bits in the input message.

2.2.3 Encryption

The encryption process consists of the following steps (**Figure 1**):

1. Choose the plaintext to be sent.

2. Replace each letter in the text with its opposite alphabetical character excluding "A" and "a" (replacement algorithm), numerical values and symbols. Then, convert the plaintext into ASCII code and then into binary code.

3. The ssDNA OTP key is generated. (The length of the key depends on (i) the length of binary plaintext and (ii) the number of bits required to represent each nucleotide.)

4. Scan the binary sequence from left to right to find the occurrences of 0 and 1 s.

   - If the first digit of binary bit is 1, then this bit is compared with last n bases of OTP key and complementary data of DNA form are produced as the encrypted message where n is the number of bits required to represent the nucleotides.

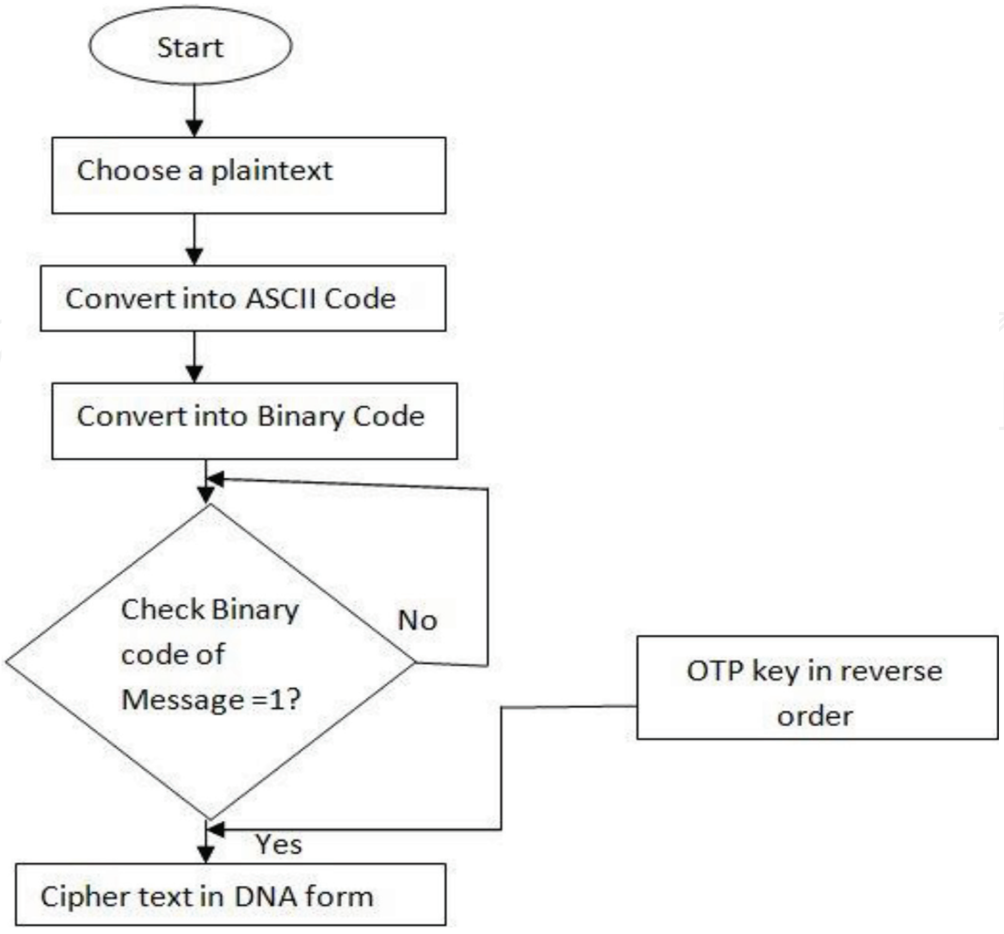


Figure 1.
Flow chart of DNA hybridization encryption.

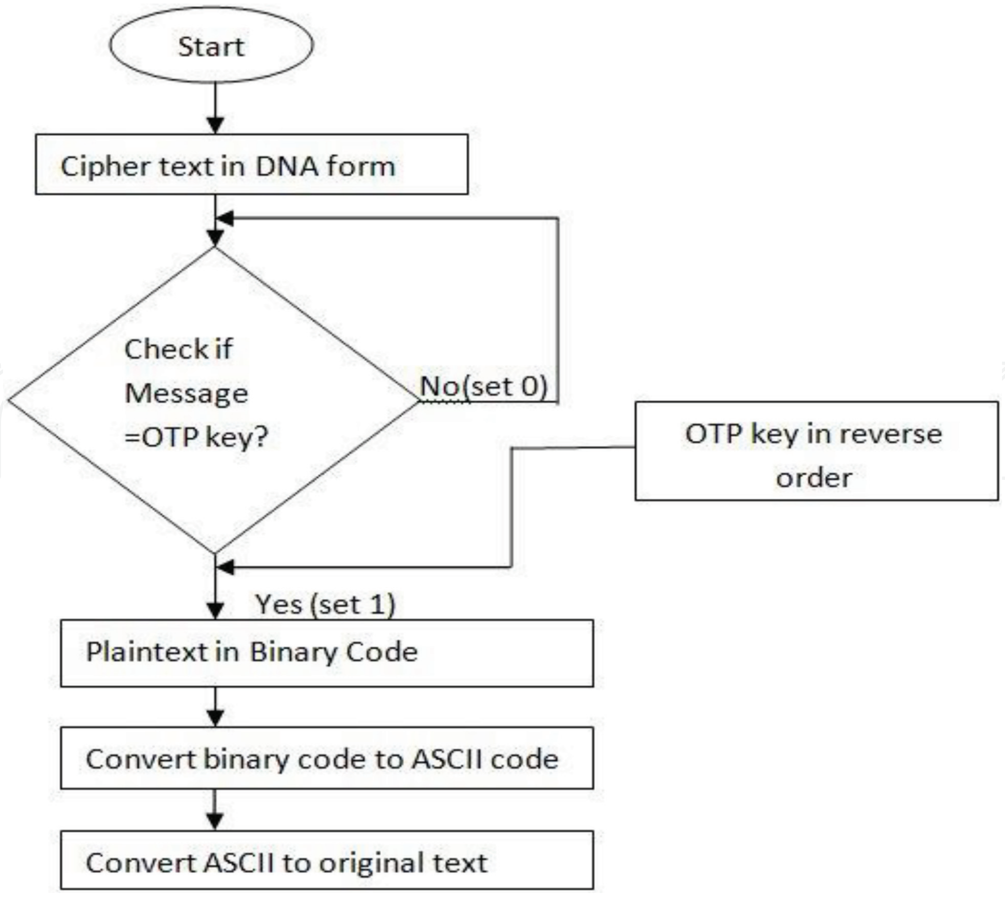


Figure 2.
Flow chart of DNA hybridization decryption.

- If the first digit of binary bit is 0, then no operation is carried out and the next n bases in the OTP key from reverse order are ignored where n is the number of bits required to represent nucleotides.

5. Repeat step 3 for all the occurrences of 1 and 0 s and put them all together to obtain the resulting ciphertext.

2.2.4 Decryption

The decryption process consists of the following steps (**Figure 2**):

1. Take n leftmost bits from the ciphertext and compare with the last n bits of the OTP key.
   - If they are found to be complementary, then binary bit "1" is formed; else, a binary "0" is formed.
2. Repeat step 1 for the subsequent n-bit sequence in the ciphertext till the end and put them all together to obtain the binary digit.
3. Apply reverse replacement algorithm.
4. Arrange the binary digits (in n bits form) and convert the value into ASCII code.
5. Convert the ASCII code to plaintext.

2.2.5 DDHO algorithm

Here, the proposed algorithm DDHO (combination of DES algorithm and DNA-based hybridization and OTP scheme) is explained in detail. First, the DES algorithm is performed on the given plaintext and key and the resultant ciphertext is taken as an input to the DNA hybridization and OTP scheme. Further, the algorithm proceeds as per the above illustrated DNA hybridization and OTP scheme.

The encryption algorithm for DDHO is as follows:

1. A block of 64 bits is permutated by an initial permutation called IP.

2. Resulting 64 bits are divided into two equal halves, each containing 32 bits, left and right halves.

3. The right half goes through a function F (Feistel function)

4. The left half is XOR-ed with output from the F function obtained in the above step.

5. The left and right halves are swapped (except the last round).

6. In the last round, apply an inverse permutation (IP-1) on both halves that is the last step which produces a ciphertext in binary form.

7. The ssDNA OTP key is generated and the length of this key depends upon the length of binary plaintext and the number of bits required to represent each nucleotide.

8. Start scanning the binary sequence, obtained in step 6, from left to right to find the occurrences of 0 and 1 s.

    - If first digit of binary bit is 1, then this bit is compared with last n bases of OTP key and complementary data of DNA form are produced as the encrypted message where n is the number of bits required to represent the nucleotides.

    - If the first digit of the binary bit is 0, then no operation is carried out and the next n bases in the OTP key from reverse order are ignored where n is the number of bits required to represent nucleotides.

9. Repeat step 8 for all the occurrences of 1 and 0 s and put them all together to obtain the resulting ciphertext.

The decryption algorithm of DDHO is the reverse process of DDHO encryption algorithm.

2.2.6 Analysis of DDHO

The plaintexts chosen for encryption and decryption using the DDHO algorithm are highly diverse. They include short and long texts, purely alphabetical text and text containing alphabets and many other characters. The plaintexts of diverse types are selected, so that they are very representative. With regard to difference of the lengths of the text, four plaintexts with increasing size are selected (**Table 1**).

The first plaintext contains only alphabetical and digital characters, the second plaintext contains only non-alphabetical and non-digital characters, and the third and the fourth plaintexts contain a combination of characters.

By applying the above test dataset to the DDHO algorithm, the original plaintext size, the resulting ciphertext size and the key size are examined, together with the encryption and decryption time. The encryption and decryption processes are performed five times for each plaintext and the average system time is obtained and listed to make the evaluation of time fair.

The results obtained are shown in the **Table 2**.

The number of bits needed to store the plaintext in ASCII format is eight times that of the length of the plaintext. For the output of DES in binary form, the number of nucleotides used to represent each bit is 10 so that the total size of key is 10 times the length of binary bits (i.e. output of DES). For instance, consider a plaintext of length 64 bits. The output of DES algorithm is also 64 bits, so the length of OTP key is equal to 64×10 bits = 640 bits. The length of ciphertext is 260 bits means that there are only 26 occurrences of number 1 in the output of DES algorithm (i.e. plaintext of DDHO) and the remaining $64 - 26 = 38$ bits are 0 s. Similarly, consider a plaintext of length 400 bits, the output of DES algorithm is equal to $400/64 = 6.25$ blocks of 64 bits. However, this length of plaintext is not an exact multiple of 64 bits. Therefore, it adds 48 bits 0 s at the end of plaintext and makes 7 blocks of 64 bit. The output of 7 blocks of 64 bits of DES is equal to 7×64 bits = 448 bits and hence the length of OTP key in DDHO algorithm is equal to 448×10 bits = 4480 bits. Similarly, there are only 102 occurrences of 1 s in the output of DES algorithm (i.e. plaintext of DDHO) and remaining bits have 0 s. The length of ciphertext depends upon how many 1 s (binary bit) are present in the plaintext. The more the number of 1 s, the more the length of the ciphertext.

As shown in **Table 2**, the ciphertext lengths are proportional to the corresponding plaintext lengths. The size of key increases hugely as the size of plaintext increases. The length of ciphertext is small as compared to the size of the

Dataset	Description
Test 1	Only alphabetical and digital characters
Test 2	Only non-alphabetical and non-digital characters
Test 3	Combination of characters
Test 4	Combination of characters

Table 1.
Plaintext of different contents for DDHO algorithm.

Dataset	Length of plaintext (bits)	Length of ciphertext (bits)	Size of key (bits)	Encryption time (milliseconds)	Decryption time (milliseconds)
Test 1	64	260	640	234	413
Test 2	400	1020	4480	244	426
Test 3	800	2330	8320	290	465
Test 4	1600	3740	16,000	517	703

Table 2.
Performance of DDHO with plaintexts of different lengths and contents.

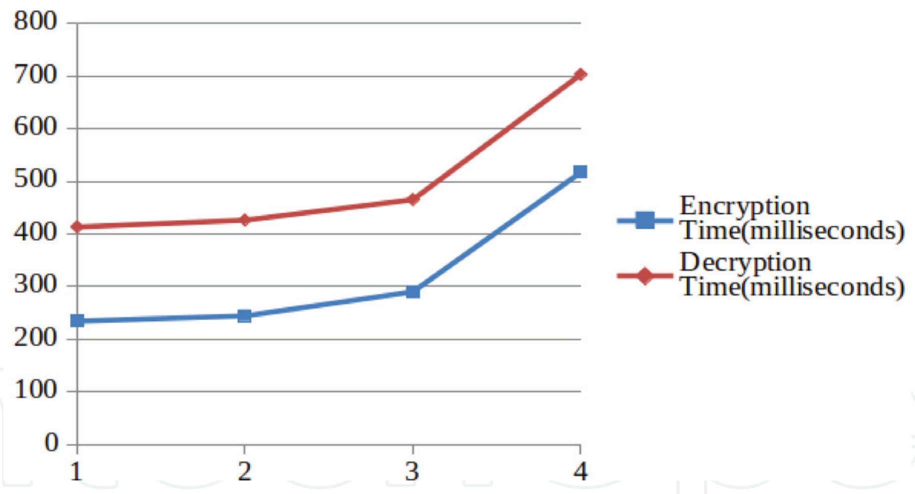


Figure 3.
Analysis of encryption and decryption times of DDHO algorithm.

key; this is because the length of ciphertext depends upon the number of 1 s present in the input plaintext.

The encryption and decryption times shown in **Table 2** and **Figure 3** show that the DDHO algorithm's encryption and decryption times for the different lengths of plaintext increase slower with the changes in the length of plaintext. This reveals that the processing time can be very fast even for relatively very long plaintext.

3. Hybrid graphical encryption

The hybrid graphical encryption algorithm (HGEA) is a unique graphical encryption algorithm based on mathematical transformations and graphical pattern realization. It is a symmetric key encryption in which a single 64-bit key is shared between two parties for encryption and decryption of data.

3.1 Hybrid cubes encryption algorithm (HiSea)

As the HGEA is inspired from hybrid cubes encryption algorithm (HiSea), it is explained here in brief.

Hybrid cubes encryption algorithm (HiSea) is the symmetric non-binary block cipher. The encryption and decryption keys, plaintext, ciphertext and internal operation in the encryption or decryption processes are based on the integer numbers. HiSea encryption algorithm was developed by Sapiee Jamel in 2011. The plaintext size for the encryption process is 64 bytes ASCII characters. Hybrid cube (HC) is generated based on the inner matrix multiplication of the layers between the two magic cubes (MCs). HC of order 4 × 4 is a matrix $H_{i,j}$, i {1, 2, 879} and j {1, 2, 3, 4}, defined as follows: $H_{i,j} = MC_{i,j} \times MC_{i+1,j}$ where the $MC_{i,j}$ is a jth layer of ith magic cube [17–19].

3.2 Hybrid graphical encryption algorithm (HGEA)

HGEA performs the operation, like in HiSea, of generating 4 × 4 matrix, then mixing it with key and again mixing of rows and column. Further, the algorithm obtains decision parameters based on remainder value and exploit the correlation between distributions of two graphical patterns for manipulation of intermediate data.

The main concept of hybrid graphical encryption algorithm cipher is to realize the input data into 8 × 8 bit matrix pattern. Then, divide it into four 4 × 4 matrices by putting it up against XY axis quadrant graph. Again, each quadrant 4 × 4 matrix is expanded into four possible 16-bit 4 × 4 matrices by XOR operation with four 4 × 4 subkeys. Further quadrant selection operation selects one 4 × 4 bit matrix output for further processing. Finally, after the final XOR operation, each set of 4 × 4 bit matrix is plotted into XY axis plot (**Figure 4**).

3.3 The encryption illustrated

The binary conversion of data and its graphical representation are the key aspects of hybrid graphical encryption algorithm cipher. The five steps involved in the encryption process are: (1) conversion and arrangement, (2) transformation, (3) selection, (4) plotting, and (5) arrangement and conversion.

3.3.1 Step 1: conversion and arrangement

This encryption algorithm can process any "n" plaintext ASCII characters from input file. The input string is split into 8 bytes of m parts. Then, the input ASCII message bit is put up against the standard ASCII table. The plaintext value is then replaced by its ASCII value according to the table. This encryption encompasses numbers, special characters and even spaces.

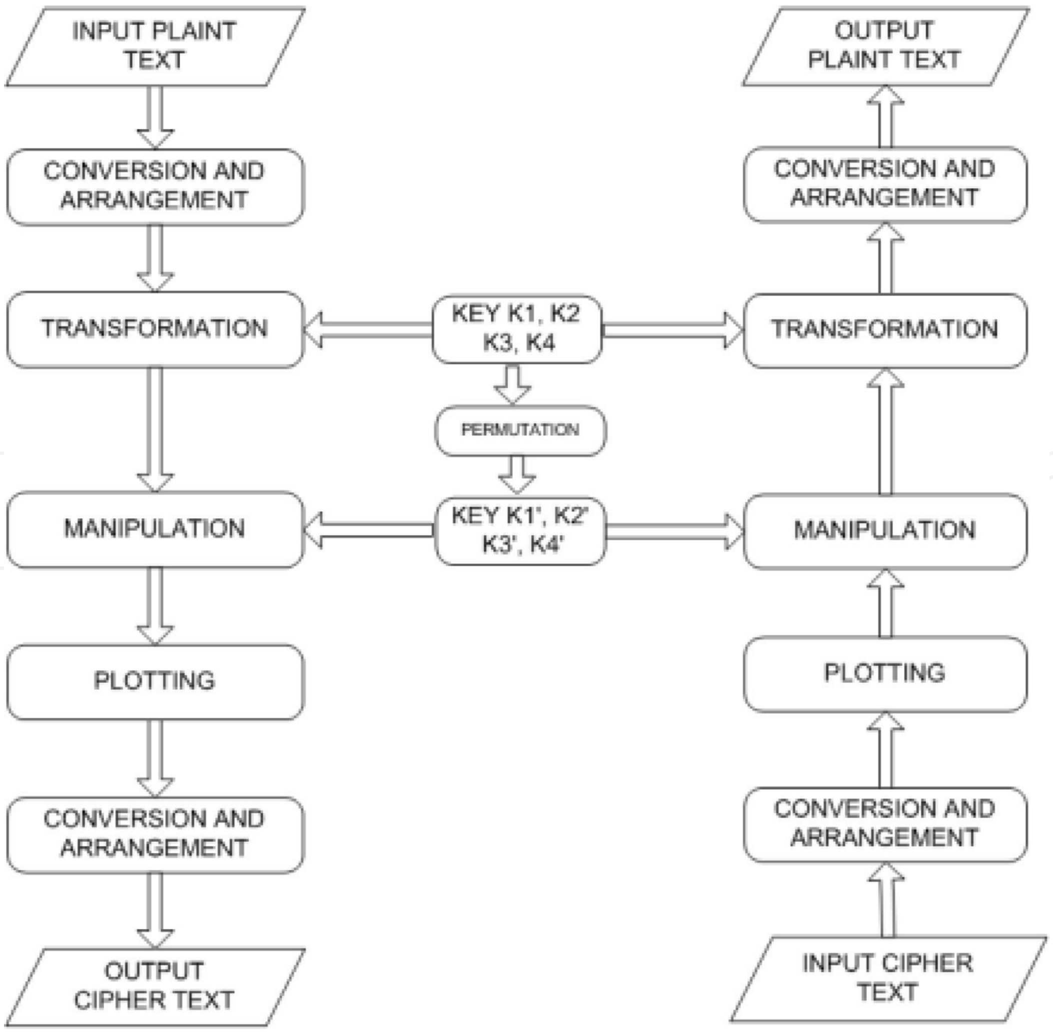


Figure 4.
System architecture diagram of HGEA.

Since this algorithm will be using the entire ASCII table for referencing, the case sensitivity of the message will play a very crucial part in output ciphertext.

After tabulating the plaintext in comparison with the ASCII table, ASCII and decimal values of the plaintext can be derived. Now, the decimal value has to be converted to binary value to move on to the next step. For binary values that do not reach the 8-bit mark, 0 s are added to the back. The obtained binary value is then tabulated in the form of an 8 × 8 matrix as shown in **Table 3**.

Since HGEA is a symmetric key encryption, a 64-bit binary key is shared for both encryption and decryption processes. These keys are also tabulated in the form of an 8 × 8 matrix of message bits.

Consider key bits as 64 random bits tabulated in an 8 × 8 matrix form, similar to message bits.

3.3.2 Step 2: transformation

In this step, initially the 8 × 8 matrix is divided into quadrant form as shown in **Table 4(a)**. The 8 × 8 matrix formed from plaintext is divided into four 4 × 4 matrices, that is, quarters named as M_1, M_2, M_3 and M_4 and generalized as M_i Similarly, the 8 × 8 matrix form key is also divided into four quarters named as K_1, K_2, K_3 and K_4, generalized as K_i.

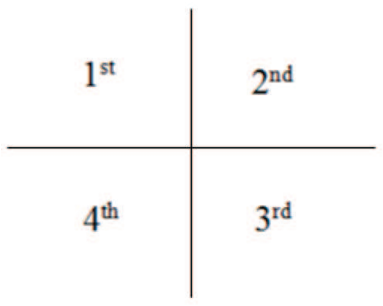
Again, M_i which is a 4 × 4 matrix is converted into 8 × 8 matrix by performing XOR operation of Mi with K_1, K_2, K_3 and K_4. M_1 is XOR-ed with K_1, K_2, K_3 and K_4 and obtained value is populated to 1st, 2nd, 3rd and 4th quadrants, respectively, as shown in **Table 4(b)**.

Plain Text	ASCII	Binary
M	77	01001101
E	69	01000101
S	83	01010011
S	83	01010011
A	65	01000001
G	71	01000111
E	69	01000101
0	0	00000000

$$M_{Plaintext} = \begin{matrix} 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \end{matrix}$$

Table 3.
Conversion of plaintext to binary.

(a)



(b)

01001101	0100	1101
01000101	0100	0101
01010011	0101	0011
01010011	0101	0011
01000001	0100	0001
01000111	0100	0111
01000101	0100	0101
00000000	0000	0000

Table 4.
Reference XY axis and sample conversion of 8 × 8 matrix into 4 × 4 matrix.

Then shifting operation is performed as follows:

For $M_1 \oplus K_1$, $M_2 \oplus K_1$, $M_3 \oplus K_1$ and $M_4 \oplus K_1$
- 1st row, $R_1 \leftarrow$ no shift
- 2nd row, $R_2 \leftarrow$ 1 bit
- 3rd row, $R_3 \leftarrow$ 2 bits
- 4th row, $R_4 \leftarrow$ 3 bits

For $M_1 \oplus K_2$, $M_2 \oplus K_2$, $M_3 \oplus K_2$ and $M_4 \oplus K_2$
- 1st row, $R_1 \leftarrow$ 3 bit
- 2nd row, $R_2 \leftarrow$ no shift
- 3rd row, $R_3 \leftarrow$ 1 bit
- 4th row, $R_4 \leftarrow$ 2 bits

For $M_1 \oplus K_3$, $M_2 \oplus K_3$, $M_3 \oplus K_3$ and $M_4 \oplus K_3$
- 1st row, $R_1 \leftarrow$ 2 bits
- 2nd row, $R_2 \leftarrow$ 3 bits
- 3rd row, $R_3 \leftarrow$ no shift
- 4th row, $R_4 \leftarrow$ 1 bit

For $M_1 \oplus K_4$, $M_2 \oplus K_4$, $M_3 \oplus K_4$ and $M_4 \oplus K_4$
- 1st row, $R_1 \leftarrow$ 1 bit
- 2nd row, $R_2 \leftarrow$ 2 bits
- 3rd row, $R_3 \leftarrow$ 3 bits
- 4th row, $R_4 \leftarrow$ no shift

3.3.3 Step 3: selection

As its name suggests, first quadrant selection operation is performed, which gives the selected quadrant value for further processing. In this step, only one 4×4 matrix value is selected for further processing for each Mi value. Thus, step. 2 and 3 via series of confusion and logical operations propose four possible 4×4 matrix values for further processing, and finally, the selection step selects only one 4×4 matrix value for further processing. For this purpose, counters are deployed, which will count the number of 1 s in each quadrant of subkeys K_1, K_2, K_3 and K_4 for M_1', M_2', M_3' and M_4', respectively. Then, the total number of 1 s in corresponding K_i is divided by 4 and the remainder is found.

$$Remainder(R_i) = \frac{Total\_no\_of\_1s\_in\_K_i}{4} \quad (1)$$

Depending upon the total number of 1 s in K_i for corresponding M_i, the selected quadrant value will be decided.

$$Q_s = R_i + 1 \quad (2)$$

For instance, let us consider that for M_1, the total number of 1 s in K_1 is calculated (consider 7, for example) and divided by 4. Now, considering the remainder which will be 3, hence $Q_s = 4$, the fourth quadrant is selected for further processing and denoted as M_{is}.

After this, we pass the K_i via permutation box "P," which will shift the bit position of standard matrix as per bit position of randomly selected transposition matrix shown in the **Table 5**.

Finally, M_i' XOR K_i' is performed and M_i" is generated. Thus, matrices M_1*, M_2*, M_3* and M_4* matrix are generated.

$$M_i^* \leftarrow M_{is} \oplus K_i^{'} \tag{3}$$

3.3.4 Step 4: plotting

Now, consider the standard matrix distribution of any 4 × 4 matrix as shown in **Table 6**. Each of the four M_1*, M_2*, M_3* and M_4* values will have different transformations when plotted in XY graph. Values of M_1" will be populated to 1st quadrant as per graph position, which can be realized by permutation as shown in **Tables 7** and **8**.

In this way, the values of M_1", M_2", M_3" and M_4" are populated to the reference XY graph.

3.3.5 Step 5: arrangement and conversion

Finally, we have M1", M2", M3" and M4" plotted in 8 × 8 matrix form. Now, each row of the matrix is converted from binay to decimal and then to plaintext

4,3	1,4	3,1	4,2
1,2	1,1	4,4	2,1
2,3	3,2	3,4	2,2
2,4	4,1	3,3	1,3

Table 5.
Reference standard 4 × 4 transposition matrix distribution.

Standard matrix distribution of 4x4 matrix

1,1	2,1	3,1	4,1
1,2	2,2	3,2	4,2
1,3	2.3	3,3	4,3
1,4	2,4	3,4	4,4

Reference standard XY graph plot

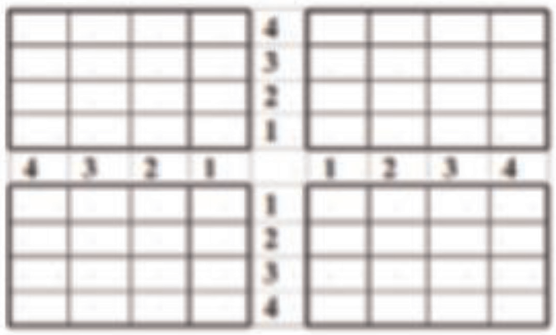


Table 6.
Plotting the values to standard XY axis graph.

P1 =

4,4	3,4	2,4	1,4
4,3	3,3	2,3	1,3
4,2	2.2	2,2	1,2
4,1	3,1	2,1	1,1

P2 =

1,4	2,4	3,4	4,4
1,3	2,3	3,3	4,3
1,2	2.2	3,2	4,2
1,1	2,1	3,1	4,1

Table 7.
Permutations P_1 and P_2 for M_1 and M_2.

P3 =

1,1	2,1	3,1	4,1
1,2	2,2	3,2	4,2
1,3	2.3	3,3	4,3
1,4	2,4	3,4	4,4

P4 =

4,1	3,1	2,1	1,1
4,2	3,2	2,2	1,2
4,3	3,3	2,3	1,3
4,4	3,4	2,4	1,4

Table 8.
Permutations P_3 and P_4 for M_3 and M_4.

Binary	ASCII	Plain Text
01001101	77	M
01000101	69	E
01010011	83	S
01010011	83	S
01000001	65	A
01000111	71	G
01000101	69	E
00000000	0	0

Table 9.
Example conversion of ASCII cipher into binary.

characters by referring the standard ASCII table, as illustrated in **Table 9**, and this is our ciphertext.

3.4 The decryption process

Decryption is also performed in the same manner as encryption but in reverse order. The steps involved in decryption are: (1) conversion and arrangement, (2) plotting, (3) selection, (4) transformation and (5) arrangement and conversion.

3.5 The algorithms

In this section, the algorithms of HGEA encryption and decryption are explained step by step. First, the encryption algorithm is given below.

Algorithm 1: **HGEA encryption**

Input: Plaintext p, which is divided into n 8-bit characters. Each 8-bit value ASCII character is converted into 64-bit binary values in 8 × 8 matrix Mp, which is the input for the algorithm. Similarly, 64-bit random key (K) is used.

Output: Ciphertext C, where C can be decrypted using key K to its corresponding plain-text P.

```
1: Mp ← a(i, j); where 0 < (i, j) < 7
2: Split 8 × 8 matrix into M_1, M_2, M_3, M_4, 4 × 4 matrix.
3: for, (i = 0; i < 4; i++);
```

```
4:     for, (j = 0; j < 4; j++);
5:         $M_1$ [i] [j] ← Mp;
6: for, (i = 0; i < 4; i++);
7:     for, (j = 0; j < 4; j++);
8:         l = j + 4;
9:         $M_2$ [i] [j] ← Mp;
10: for, (i = 0; i < 4; i++);
11:     k = i + 4:
12:     for, (j = 0; j < 4; j++);
13:         $M_3$ [i] [j] ← Mp;
14: for, (i = 0; i < 4; i++);
15:     k = i + 4:
16: for, (j = 0; j < 4; j++);
17:     l = j + 4;
18:     $M_4$ [i] [j] ← Mp;
19: Generate $K_1$, $K_2$, $K_3$, $K_4$ from K similar to plaint text.
20: Compute:
21:     $M_1 \oplus K_1$, $M_1 \oplus K_2$, $M_1 \oplus K_3$, $M_1 \oplus K_4$;
22:     $M_2 \oplus K_1$, $M_2 \oplus K_2$, $M_2 \oplus K_3$, $M_2 \oplus K_4$;
23:     $M_3 \oplus K_1$, $M_3 \oplus K_2$, $M_3 \oplus K_3$, $M_3 \oplus K_4$;
24:     $M_4 \oplus K_1$, $M_4 \oplus K_2$, $M_4 \oplus K_3$ and $M_4 \oplus K_4$;
25: Perform Right Shift Operation circularly:
26: With $M_1 \oplus K_1$, $M_2 \oplus K_1$, $M_3 \oplus K_1$ and $M_4 \oplus K_1$
27:     1st row no shift
28:     Right circular shift the 2nd row by 1 bit
29:     Right circular shift the 3rd row by 2 bits
30:     Right circular shift the 4th row by 3 bits
31: With, $M_1 \oplus K_2$, $M_2 \oplus K_2$, $M_3 \oplus K_2$ and $M_4 \oplus K_2$
32:     Right circular shift the 1st row by 3 bits
33:     2nd row no shift
34:     Right circular shift the 3rd row by 1 bit
35:     Right circular shift the 4th row by 2 bits
36: With $M_1 \oplus K_3$, $M_2 \oplus K_3$, $M_3 \oplus K_3$ and $M_4 \oplus K_3$
37:     Right circular shift the 1st row by 2 bits
38:     Right circular shift the 2nd row by 3 bits
39:     3rd row no shift
40:     Right circular shift the 4th row by 1 bit
41: With $M_1 \oplus K_4$, $M_2 \oplus K_4$, $M_3 \oplus K_4$ and $M_4 \oplus K_4$
42:     Right circular shift the 1st row by 1 bit
43:     Right circular shift the 2nd row by 2 bits
44:     Right circular shift the 3rd row by 3 bits
```

```
45:    4th row no shift
46: Compute selected quadrant value ($Q_s$):
47:    Remainder ($R_{is}$) ← ∑1's in $K_i$/4, for corresponding $M_i$ respectively.
48:    For $Q_{is}$ = $R_i$ + 1.
49: Generate $K_1$', $K_2$', $K_3$' and $K_4$'
50:    $K_i$' = $\pi_i(K_i)$,where π = fixed permutation Table.
51: Compute: $M_i$* ← $M_{is}$ ⊕ $K_i$'
52:    $M_i$ cipher = π (Mi*)
53:    M cipher ← ($M_1$cipher + $M_2$cipher) ($M_4$cipher + $M_3$cipher)
54: Convert each row of 8 × 8 matrix binary into ciphertext
```

Next, the decryption algorithm is given below.

Algorithm 2: **HGEA decryption**

Input: Ciphertext C which is divided into n 8-bit characters. Each 8-bit value ASCII character is converted into 64-bit binary values in 8 × 8 matrix M_c, which is the input for the algorithm. Similarly, 64 bit random key (K) is used.

Output: Plaintext P, where P is decrypted to plaintext using key K.

```
1: Mc ← a(i, j); where 0 < (i, j) < 7
2: Split 8 × 8 matrix into $M_1$", $M_2$", $M_3$", $M_4$" 4 × 4 matrix.
3:    for, (i = 0; i < 4; i++);
4:        for, (j = 0; j < 4; j++);
5:        $M_1$" [i] [j] ← Mc;
6: for, (i = 0; i < 4; i++);
7:    for, (j = 0; j < 4; j++);
8:        i = j + 4;
9:        $M_2$" [i] [j] ← Mc;
10: for, (i = 0; i < 4; i++);
11:        k = i + 4:
12:      for, (j = 0; j < 4; j++);
13:          $M_3$" [i] [j] ← Mc;
14: for, (i = 0; i < 4; i++);
15:    k = i + 4:
16:    for, (j = 0; j < 4; j++);
17:        l = j + 4;
18:        $M_4$" [i] [j] ← Mp;
19: Generate $K_1$, $K_2$, $K_3$, $K_4$ from K similar to plaint text.
20:    $M_i$ cipher = π ($M_i$*)
21: Generate $K_1$', $K_2$', $K_3$' and $K_4$'
22:    $K_i$' = $\pi_i(K_i)$,where π = fixed permutation table.
23: Compute: $M_i$* ← $M_{is}$ ⊕ $K_i$'
24: Compute selected quadrant value ($Q_s$):
```

```
25:     Remainder(R) ← $\sum$1's in $K_i$ / 4, for corresponding $M_i$ respectively.
26:     For $R_i$ = n, $Q_{is}$ = n + 1.
27: Perform Right Shift Operation circularly:
28: If $Q_s$ = 1
29:     1st row no shift
30:     Left circular shift the 2nd row by 1 bit
31:     Left circular shift the 3rd row by 2 bits
32:     Left circular shift the 4th row by 3 bits
33: If $Q_s$ = 2
34:     Left circular shift the 1st row by 3 bits
35:     2nd row no shift
36:     Left circular shift the 3rd row by 1 bit
35:     Left circular shift the 4th row by 2 bits
37: If $Q_s$ = 3
38:     Left circular shift the 1st row by 2 bits
39:     Left circular shift the 2nd row by 3 bits
40:     3rd row no shift
41:     Left circular shift the 4th row by 1 bit
42: if $Q_s$ = 4
43:     Left circular shift the 4th row by 1 bit
44:     Left circular shift the 4th row by 2 bits
45:     Left circular shift the 4th row by 3 bits
46:     4th row no shift
47: Selected quadrant value ($K_{is}$) ← $Q_s$
48: $M_i$ ← $M_i$' $\oplus$ $K_{is}$
49: Generate $M_1$, $M_2$, $M_3$ and $M_4$.
50: Mp ← ($M_1$cipher + $M_2$cipher) ($M_4$cipher + $M_3$cipher)
51: Convert above result to string to get plaintext.
```

3.6 Performance evaluation

Performance measurement criterion is the time taken by the algorithms to perform encryption and decryption of the input text file, that is, the encryption computation time and decryption computation time. [20]

3.6.1 Computation time for encryption and decryption

The encryption computation time of the encryption algorithm is the time taken by the algorithm to produce the ciphertext from the plaintext. The encryption time can be used to calculate the encryption throughput of the algorithms.

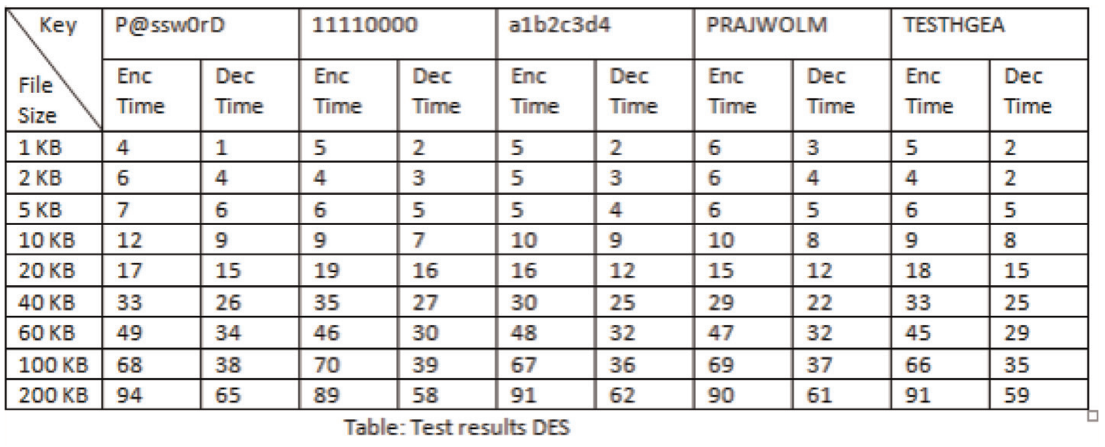
The decryption computation time is the time taken by the algorithms to produce the plaintext from the ciphertext. The decryption time can be used to calculate the decryption throughput of the algorithms.

For evaluation, files of sizes 1, 2, 5, 10, 20, 40, 60, 100 and 200 KB were used as input data files. For the sake of comparison, same input files have been set as input to both HGEA and DES. [14] DES has also been implemented in the same environment, JAVA version 8 and simulated in IntelliJ IDEA v.2018.1.4 on the same machine (**Tables 10** and **11**).

The same data set was encrypted via both DES and HGEA algorithms. Each sample data set was encrypted with five randomly chosen passwords, and the encryption execution time and decryption execution time were listed as shown in **Table 12**.

The whole test was performed very carefully. During the test, it was observed that sometimes, when new input data are fed into the computation model, the test results returned false encryption and decryption time values; for example, the test model resulted in very high encryption and decryption execution time values or the test model's decryption execution time was very high compared to encryption computation time and the decrypted plaintext differed from input plaintext. So to resolve this issue, a new terminal was generated in Intellij whenever the input value was changed.

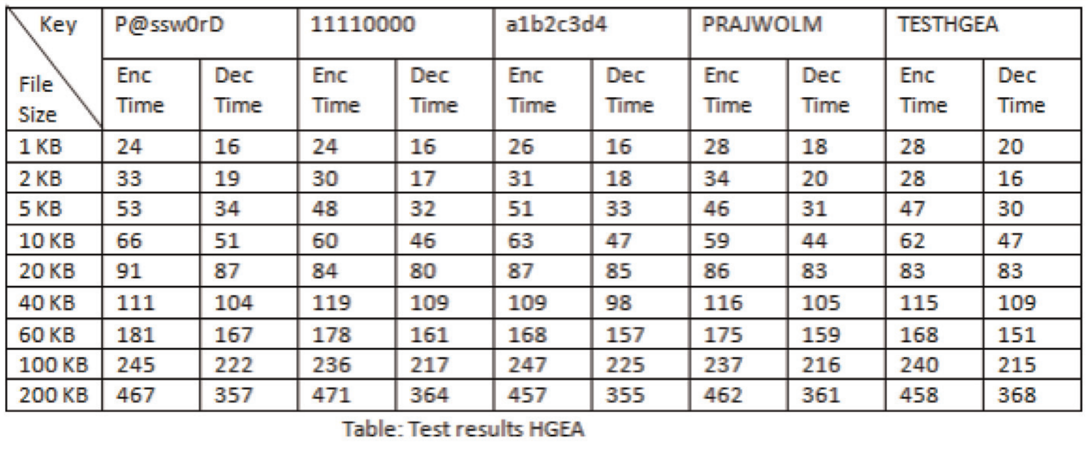
The average values of execution time for encryption and decryption were computed and are tabulated in **Tables 12** and **13**.



Key / File Size	P@ssw0rD		11110000		a1b2c3d4		PRAJWOLM		TESTHGEA	
	Enc Time	Dec Time	Enc Time	Dec Time	Enc Time	Dec Time	Enc Time	Dec Time	Enc Time	Dec Time
1 KB	4	1	5	2	5	2	6	3	5	2
2 KB	6	4	4	3	5	3	6	4	4	2
5 KB	7	6	6	5	5	4	6	5	6	5
10 KB	12	9	9	7	10	9	10	8	9	8
20 KB	17	15	19	16	16	12	15	12	18	15
40 KB	33	26	35	27	30	25	29	22	33	25
60 KB	49	34	46	30	48	32	47	32	45	29
100 KB	68	38	70	39	67	36	69	37	66	35
200 KB	94	65	89	58	91	62	90	61	91	59

Table: Test results DES

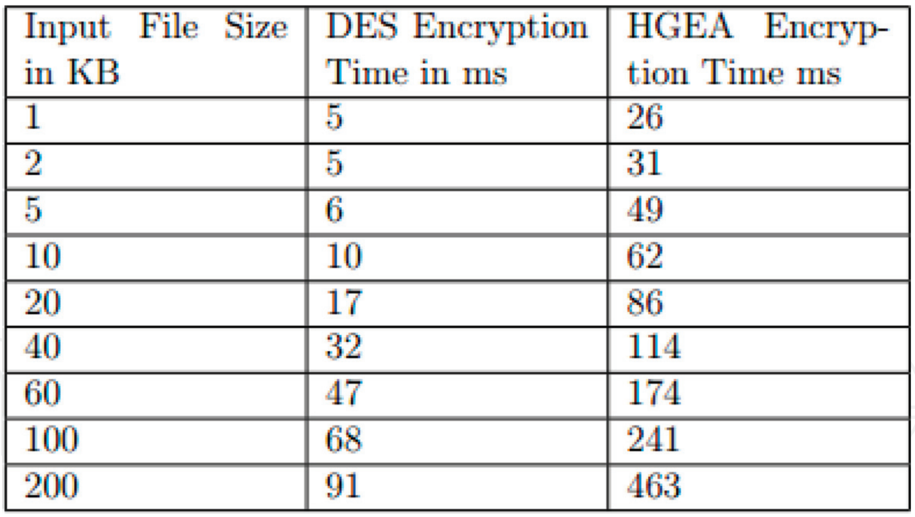
Table 10.
Execution time of DES.



Key / File Size	P@ssw0rD		11110000		a1b2c3d4		PRAJWOLM		TESTHGEA	
	Enc Time	Dec Time	Enc Time	Dec Time	Enc Time	Dec Time	Enc Time	Dec Time	Enc Time	Dec Time
1 KB	24	16	24	16	26	16	28	18	28	20
2 KB	33	19	30	17	31	18	34	20	28	16
5 KB	53	34	48	32	51	33	46	31	47	30
10 KB	66	51	60	46	63	47	59	44	62	47
20 KB	91	87	84	80	87	85	86	83	83	83
40 KB	111	104	119	109	109	98	116	105	115	109
60 KB	181	167	178	161	168	157	175	159	168	151
100 KB	245	222	236	217	247	225	237	216	240	215
200 KB	467	357	471	364	457	355	462	361	458	368

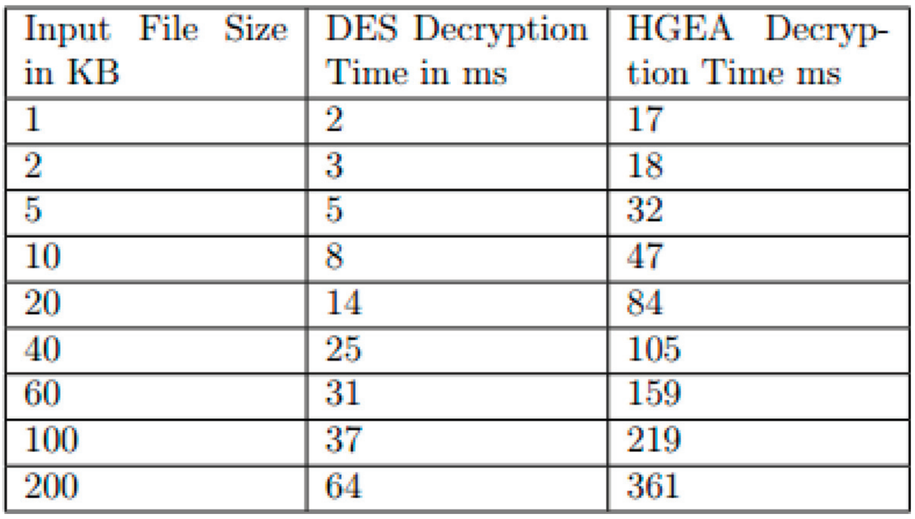
Table: Test results HGEA

Table 11.
Execution time of HGES.



Input File Size in KB	DES Encryption Time in ms	HGEA Encryption Time ms
1	5	26
2	5	31
5	6	49
10	10	62
20	17	86
40	32	114
60	47	174
100	68	241
200	91	463

Table 12.
Computation time of encryption for various input sizes.



Input File Size in KB	DES Decryption Time in ms	HGEA Decryption Time ms
1	2	17
2	3	18
5	5	32
10	8	47
20	14	84
40	25	105
60	31	159
100	37	219
200	64	361

Table 13.
Computation time of decryption for various input size.

The above tabulated values represent the computation time of various sizes of sample data sets being processed by DES and HGEA. Results show that for the proposed HGEA, execution time is 26, 49 and 62 for 1, 5 and 10 KB of data, respectively. HGEA has a much slower execution time compared to DES. The execution time of the proposed algorithm increases with an increase in data size.

Further, by using the values of **Tables 12** and **13**, various graphs showing the encryption execution and decryption execution times for DES and HGEA with different input sizes were generated as shown in **Figure 5(a)–(c)**.

The above graph shows that the throughput of DES is considerably better than HGEA for small-size input files and it increases with an increase in file size compared to HGEA. It also shows that encryption time for both algorithms is more than decryption time.

The variation in encryption time and decryption time in HGEA is due to steps followed during decryption. During encryption process in the transformation step,

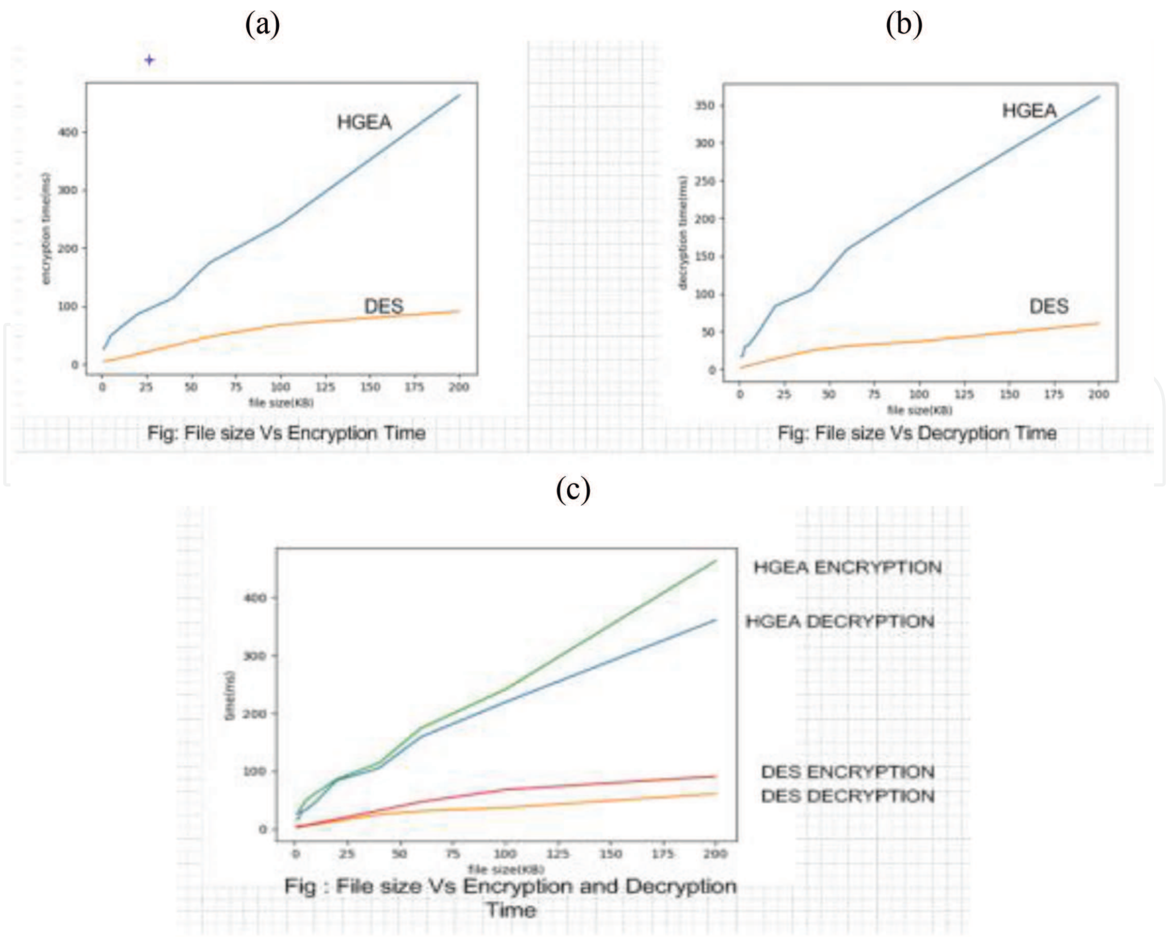


Figure 5.
File size vs. execution time of DES and HGEA.

each Mi is XOR-ed with all of the possible subkeys and only one of four possible 4×4 matrices is chosen for further processing. However, during decryption in the transformation step, only one 4×4 matrix is directly generated depending upon selected quadrant value. So, the generation of all of the possible intermediate values is one of the reasons for variation of encryption and decryption times in HGEA.

4. Conclusion

The DNA cryptosystem containing DNA hybridization technique and secure key generation technique OTP are studied, explained and implemented by taking input from the DES algorithm output. The output of DES-based DNA cryptography algorithm has encrypted message in the form of DNA sequences and the decrypted message is the original plaintext.

The DDHO algorithm is tested on different types of plaintext; the encryption and decryption times are calculated; the analysis of length of plaintext, length of ciphertext and size of key is done and found that the length of ciphertext is proportional to the corresponding plaintext length. The encryption and decryption times increase slower with the changes in the length of plaintext.

From the process of analyzing various cryptographic algorithms, a unique encryption algorithm "hybrid graphical encryption algorithm" has been proposed. The algorithm was based on hybrid cubes encryption algorithm (HiSea). The features like graphical interpretation and computation of selected quadrant value are the unique features of this algorithm, which is different from existing standard encryption algorithms. Also, the multiple transformation, multiple key generation provides, combined with graphical interpretation provided added security to the algorithm. The realization of computation model proved the proposed encryption algorithm is practically realizable. Further comparative analysis of computation time of realized model was made.

Although the comparative analysis of proposed model is proven to be more secure, the model was slower than DES. However, the processing units of modern day computer system are extremely high and developing rapidly, the implementation of HGEA possible.

Author details

Roshan Chitrakar[1,2]*, Roshan Bhusal[2] and Prajwol Maharjan[2]

1 Nepal Open University, Nepal

2 Nepal College of Information Technology, Nepal

*Address all correspondence to: roshanchi@gmail.com

References

[1] Hershey JE. Cryptography Demystified. 1st ed. New York: McGraw-Hill Telecom; 2003. pp. 4-4

[2] Pramanik S, Kumar SS. DNA Cryptography, 7th IEEE Conference2012. pp. 551-554

[3] Tausif A, Kumar A, Sanchita P. DNA cryptography based on symmetric key exchange. International Journal of Engineering and Technology. 2015;**7**(3)

[4] Jie C. A DNA based bio molecular cryptography design. In: Proceedings of IEEE International Symposium. Vol. 3. 2003. pp. 777-822

[5] Anupriya A, Praveen K. Secure data transmission using DNA encryption. International Journal of Advanced Research in Computer Science. 2014; **5**(7):57-61

[6] Shreyas C. DNA cryptography based on DNA hybridization and one time pad scheme. International Journal of Research and Technology. 2013;**2**(10)

[7] Aashish G, Thomas L, John R. DNA based cryptography. In: 5th DIMACS Workshop on DNA Based Computers. MIT; 1999

[8] Jamel S et al. The hybrid cubes encryption algorithm (HiSea). In: Advances in Wireless, Mobile Networks and Applications. Berlin, Heidelberg: Springer; 2011. pp. 191-200

[9] Jamel S, Deris MM, Tri I, Yanto R, Herawan T. HiSea: A non binary toy cipher. Journal of Computing. 2011; **3**(6):2027

[10] Nidhi G, Sanjay K. Pseudo DNA cryptography technique using OTP key for secure data transfer. International Journal of Engineering Science and Computing. 2016;**6**(5)

[11] Sridevi R, Karthika S. Secured image transfer through DNA cryptography using symmetric cryptographic algorithm. International Journal of Engineering Research and Technology. 2013;**2**(2)

[12] Habibulla K, Srikanth M, et al. FGPA implementation of DES algorithm using DNA cryptography. Journal of Theoretical and Applied Information Technology. 2017;**95**(10):2147-2158

[13] Ravi G. DNA-based cryptography approaches using central dogma of molecular biology, Thesis. Patiala: Computer Science and Engineering Department, Thapar University; 2014

[14] Hung-wei H. Solving the travelling salesman problem by ant colony optimization algorithms with DNA computing, Master Thesis; 2004

[15] Singh S, Kumar D. Secret data writing using DNA sequences. In: IEEE International Conference. 2011. pp. 402-428

[16] Stemmer WPC, Crameri A. DNA Mutagenesis by Random Fragmentation and Reassembly. U.S. Patent No. 5, 830, 721.3 Nov. 1998

[17] Mushtaq MF, Jamel S, Deris MM. Triangular coordinate extraction (TCE) for hybrid cubes. Journal of Engineering and Applied Sciences. 2017;**12**(8):2164-2169

[18] Jamel S, Deris MM, Yanto ITR, Herawan T. The hybrid cubes encryption algorithm (HiSea). In: Communications in Computer and Information Science. Vol. 154. Berlin Heidelberg: Springer-Verlag; 2011. p. 191200

[19] Mushtaq MF, Jamel S, Mohamad KM, Khalid SAA, Deris MM. Key

generation technique based on triangular coordinate extraction for hybrid cubes. Journal of Telecommunication, Electronic and Computer Engineering. 2017;**9**(3-4): 195-200

[20] Singh V, Dhiman H, Khatkar M. A comprehensive study of time complexity of various encryption algorithm. International Journal of Advances in Engineering and Technology. 2014;**7**(2):496