Image Encryption Schemes: A Complete Survey

Yamini Jain\textsuperscript{1}, Ritesh Bansal\textsuperscript{2}, Gaurav Sharma\textsuperscript{3}, Bhuvnesh Kumar\textsuperscript{4} and Shailender Gupta\textsuperscript{5}

\textsuperscript{1} YMCA University of Science and Technology, Faridabad
\textsuperscript{2} YMCA University of Science and Technology, Faridabad
\textsuperscript{3} YMCA University of Science and Technology, Faridabad
\textsuperscript{4} YMCA University of Science and Technology, Faridabad
\textsuperscript{5} YMCA University of Science and Technology, Faridabad

\texttt{jain.yamin34@gmail.com}, \texttt{ritesh.bansal@hotmail.com},
\texttt{sharma.grv69@gmail.com}, \texttt{singh.bhuvneshkmr@gmail.com}, \texttt{shailender81@gmail.com}

Abstract

Advancement in digital technologies has resulted in increased data transfer over internet in recent years. As a result, security of images/data is one of the biggest concern of many researchers. Therefore several cryptographic schemes have been proposed for image/data encryption. An efficient cryptographic scheme is one that have high brute force search time, low execution time complexity and should be able to provide good security. In this paper, several ciphers (traditional as well as modern) for images are compared based on various parameters such as: Time complexity, Peak Signal to Noise Ratio (PSNR), Number of Pixels Change Rate (NPCR), Unified Average Changing Intensity (UACI) and Entropy. In addition, the paper also shows the shortcomings of traditional ciphers that were used for text and how modern ciphers overcome this limitations. The analysis of simulation result shows that chaotic encryption schemes are most efficient and better than others.

Keywords: Entropy, Key Sensitivity, Chaotic, Correlation.

1. Introduction

Increased use in communication of images and data over internet has also enhanced risk of information leakage. Therefore, need for secure data transmission has risen. Cryptography [1-2] is one mechanism that provides confidentiality of data and thus ensures data security. In this mechanism, the image/data to be sent is transformed into another form known as cipher text making it difficult for the intruder to read. The reverse process of transformation known as decryption is carried out at the receiver side to recover the plain text. The cryptography mechanisms can be broadly classified into two categories:

- **Symmetric key Cryptography**: The same key will be used for encryption and decryption of images/data.
- **Asymmetric key Cryptography**: In this mechanism, two types of keys are used: Public key (known to all) and Private Key (known to intended user). The sender encrypts the data using public key of the destination and the receiver decrypts it via its private key. This mechanism provides much better security in comparison to symmetric key cryptography but at the cost of high time complexity.
As aforementioned, symmetric key mechanism provides high speed for encryption, therefore are popularly used for image encryption. In the early stages of image encryption, traditional ciphers like DES [7-13], AES [6-14], 3-DES [9-19] and several others were used but were a failure due to the reasons mentioned below:

- Strong correlation among adjacent pixels therefore doesn’t provide good security
- High computational time due to huge amount of data present in images.
- Low speed of execution.

Thus, researchers focused their attention on other techniques that could not only provide good randomness between plain and encrypted image but at the same time should have large key space thus ensures high brute force search time. Also, the processing time should be as low as possible. In this paper, all these things are taken into consideration for comparing various ciphers available in literature. The rest of the paper is organized as follows. Section 2 provides the details of various techniques used for comparison. Section 3 deals with simulation setup parameters while section 4 describes performance metrics. Results are presented and compared in section 5. Section 6 provides the concluding remarks followed by references.

2. Techniques Used for Comparison

Various techniques implemented in this paper have been described in this section.

2.1. Data Encryption Standard (DES)

![Figure 1. Single Round of DES](image)
The Data Encryption Standard is a symmetric-key algorithm used for the encryption of electronic data. It was developed in the early 1970s at IBM and is based on an earlier design by Horst Feistel. DES is based on a cipher known as the Feistel block cipher. It consists of a number of rounds where each round contains bit-shuffling, non-linear substitutions (S-boxes) and exclusive OR operations. DES is therefore a symmetric, 64 bit block cipher as it uses the same key for both encryption and decryption and operates on 64 bit blocks of data at a time. In image encryption, the data is divided into blocks of 64 bits and then DES is applied to each block. The key size used is 56 bits, however a 64 bit (or eight-byte) key is actually input. The least significant bit of each byte like every 8th, 16th, 24th, 32nd etc. bit is either used for parity (odd for DES) or is set arbitrarily and does not increase the security by any means. If the number of bits in the message is not evenly divisible by 64, then the last block will be padded with extra zeroes. Multiple permutations and substitutions are assimilated throughout in order to increase the difficulty of performing a cryptanalysis on the cipher. DES algorithm consists of 16
identical rounds as shown by figure 2 and figure 3. Figure 1 shows the block diagram of a single round. The detail of each round is given below:

1. **The E-box expansion permutation**: Here the 32-bit input data from \( R_{i-1} \) is expanded and permuted to give the 48 bits necessary for combination (ex-or) with the 48 bit key.

2. The bit by bit addition modulo 2 (or exclusive OR) of the E-box output and 48 bit sub-key \( K_i \).

3. **The S-box substitution** - This is a highly important substitution which accepts a 48-bit input and outputs a 32-bit number. The S-boxes are the only non-linear operation in DES and are therefore the most important part of its security. The input to the S-boxes is 48 bits long arranged into 8, 6 bit blocks. There are 8 S-boxes each of which accepts one of the 6 bit blocks. The output of each S-box is a four bit number.

4. **The P-box permutation** - This simply permutes the output of the S-box without changing the size of the data. It has a one to one mapping of its input to its output giving a 32 bit output from a 32 bit input.

The detailed algorithm of DES encryption is given below:

---

**Algorithm for DES Cryptography (Sender Side)**

**Key Generation:**

1. Select a 64 bit key. Convert it into a 56 bit key using permutation.

2. Split the 56 bit key into two equal parts of 28 bits each and name them \( C_0 \) and \( D_0 \).

3. Perform left circular shift operation of one bit on \( C_0 \) and \( D_0 \) to obtain \( C_1 \) and \( D_1 \).

   \[
   C_i = \text{circshift} (C_{i-1}, [0, -(1)]); \quad \text{for left shift}
   
   D_i = \text{circshift} (D_{i-1}, [0, -(1)]);
   \]

4. Combine \( C_1 \) and \( D_1 \) to get 56 bit key. Apply permutation to compress this key to 48 bits and call it key \( K_1 \).

5. Similarly obtain all 16 (48 bit) subkeys, \( K_1 \) to \( K_{16} \).

**Image Encryption:**

1. Obtain the image and convert it into grey image.

2. Divide it into sub blocks of 64 bits each and perform the following steps on each block.

3. Split the 64 bit data into two equal parts of 32 bits each and name them \( L_0 \) and \( R_0 \).

4. 16 rounds are used to calculate \( L_{16} \) and \( R_{16} \) from \( L_0 \) and \( R_0 \).

   \[
   \text{for } (i=1 \text{ to } 16)
   
   L_i = R_{i-1} \quad \text{and } R_i = L_{i-1} + \text{function} (K_i, R_{i-1})
   \]

5. The ‘function’ comprises of three steps.

   a) First compress 56 bit \( R_{i-1} \) block to a 48 bit block by permutation. Apply \( K_i + R_i \).

   Data = \( b_1 b_2 b_3 b_4 b_5 b_6 b_7 b_8 \).

   b) Apply S-box permutation on each group to convert it from 6 bits to 4 bits each.

   \[
   i = \text{binary to decimal} \left( \left( b_1 (1) b_1 (6) \right) \right);
   
   j = \text{binary to decimal} \left( \left( b_1 (2) b_1 (3) b_1 (4) b_1 (5) \right) \right);
   
   S_i b_j = s_i (i, j);
   
   S_i b_j = \text{decimal to binary} \left( S_i b_j, 4 \right);
   \]

   end

   data = \( S_i b_1 S_i b_2 S_i b_3 S_i b_4 S_i b_5 S_i b_6 S_i b_7 S_i b_8 \).

   As each \( Sb \) combination is of 4 bits, length of data becomes 32 bits.
6. Obtain last block of 64 bits \((L_{16} \text{ and } R_{16})\) as cipher text. Similarly, obtain all cipher blocks and combine them all to form cipher image.

Decryption of DES is same as encryption. DES algorithm is applied on cipher text with subkeys used in the reverse order as shown in the figure 3. Decrypted output or plain text is obtained as an output.

Algorithm for DES Cryptography (Receiver Side)

1. Obtain cipher image. Divide it into blocks of 64 bits after converting it into gray image.
2. Apply same steps like encryption but on the cipher image.
3. Take the same subkeys but in reverse order.
4. Obtain plain image as the result.

2.2 3DES (Triple DES) Cryptography

Triple DES (3 DES) was incorporated as a part of Data Encryption Standard in 1999. 3DES uses three different keys and three DES executions. The effective key length is 168 bits as it uses three distinct keys. The function follows an encrypt-decrypt-encrypt sequence. Figure 4 and figure 5 shows the block diagram of 3DES encryption and decryption. The DES algorithm is explained above.

![Figure 4. 3-DES Encryption](image)

![Figure 5. 3-DES Decryption](image)
Algorithm for Triple DES encryption is décried below:

**Algorithm for Triple DES Cryptography (Sender Side)**

1. Obtain the image and select three different keys of 64 bits each.
2. Follow the steps below using three different keys.
   
   \[ \text{[row col]} = \text{size (image)}; \]
   
   \[
   \text{for } i=1: \text{row} \\
   \quad \text{for } j=1:8: \text{col} \\
   \quad \quad \text{im\_block} = (a(i,j:j+7)); \\
   \quad \quad \text{output\_1} = \text{des\_encryption (key1, im\_block);} \\
   \quad \quad \text{output\_2} = \text{des\_decryption (key2, output\_1);} \\
   \quad \quad \text{output\_3} = \text{des\_encryption (key3, output\_2);} \\
   \quad \text{cipher (i, j:j+7)} = \text{output\_3} \\
   \text{end} \\
   \text{end} \\
   
   3. Get the cipher image as a result.

Decryption is just reverse of encryption and the various steps in detail are mentioned below:

**Algorithm for Triple DES Cryptography (Receiver Side)**

1. Obtain the encrypted image as 'encr\_im'.
2. Apply DES encryption and decryption on ciphered image in the following sequence.
   
   \[
   \text{for } i=1: \text{row} \\
   \quad \text{for } j=1:8: \text{col} \\
   \quad \quad \text{encr\_block} = (\text{encr\_im (i,j:j+7)}); \\
   \quad \quad \text{output\_1} = \text{des\_decryption (key}_3, \text{encr\_block);} \\
   \quad \quad \text{output\_2} = \text{des\_encryption (key}_2, \text{output\_1);} \\
   \quad \quad \text{output\_3} = \text{des\_decryption (key}_1, \text{output\_2);} \\
   \quad \text{original (i,j:j+7)} = \text{output\_3} \\
   \text{end} \\
   \text{end} \\
   
   3. The result obtained is the original image.

### 2.3 AES (Advanced Encryption Standard) Cryptography

The Advanced Encryption Standard (AES) [6, 20] is a specification for the encryption of electronic data established by the U.S. National Institute of Standards and Technology (NIST) in 2001. AES, based on a design principle known as a substitution-permutation network, is a combination of both substitution and permutation, and is fast in both software and hardware. Unlike its predecessor DES, AES does not use a Feistel network. AES is a symmetric key algorithm with key length of 128 bits. For image encryption, data is divided into 64 bits sub blocks and the technique is applied. AES has 10 rounds of complex operations that are performed on the plain block of 64 bits to get the cipher text of same length. Figure 6 and figure 7 shows the overall structure of AES encryption and Decryption.
Each round requires four operations described below:

1. **Sub Bytes**

   This operation is a simple substitution that converts every byte into a different value. AES defines a table of 256 values for the substitution. The contents of the substitution table are not arbitrary, the entries are calculated using a mathematical formula but mostly implementations simply have the substitution table stored in memory.

2. **Shift Rows**

   As the name suggests, Shift Rows operates on each row of the state array. Each row is rotated to the right by a certain number of bytes. The 16 bytes are first converted to a 4X4 matrix. Each row of the matrix is then separately circularly shifted through (row-1) times to the right.

3. **Mix Columns**

   This operation is the most challenging, both to explain and to perform. Each column of the state array is processed separately to produce a new column. The new column replaces the old one. The processing involves a matrix multiplication. The data is arranged in a 4 row matrix, as arranged in shift row. Process consists of two parts, first part explains the how the multiplication is to be done with a fixed polynomial. We have a multiplication matrix, for the multiplication step. The multiplication is done using one column of data matrix at a time. The result of the multiplication is simply the result of a lookup of the L table, followed by the addition of the results, followed by a look up to the E table.

4. **Add Round Key**

   After the Mix Columns operation, the Add Round Key is very simple indeed and hardly needs its own name. This operation simply takes the existing state array, XORs the value of the appropriate round key, and replaces the state array with the result. It is done once before the rounds start and then once per round, using each of the round keys in turn.
Round 10 is a bit different from others as it doesn’t contain the Mix Column step. Figure 8 shows the block diagram of a single round of AES encryption algorithm.

![Figure 8. Single Round of AES](image)

AES encryption is performed on a data block of 16 bytes with an expanded key of size 176 bytes. The data transforming functions, Add_round_key(), Sub_bytes(), Shift_Row(), Mix_Column() are applied in a particular manner:

1. The first transformation has 1:16 elements of the key along with the data block input in Add_round_key() function.

2. The changed state of data is then processed in a loop utilizing key elements from 17:160 and forwarding the data in the sequence Sub_bytes() -> Shift_Row() -> Mix_Column() -> Add_round_key(), with Add_round_key() taking the changed state from Mix_Column() and next 16 bytes of key as inputs. No other function uses key.

3. After the 9 rounds of transformation, the data is again forwarded in the same sequence but excluding Mix_column() from that process. The last 16 key bytes are thus employed in Add_round_key() for the last time, and the resulting block is the encrypted data block.

For the implementation of AES various transformation functions have been used which are described below:

**Transformation functions used in key expansion:**

**Rot word:**
This function performs same as Shift Row. It shifts the data bytes circularly.

**Sub Word:**
It is same as Byte sub. A same s-box table is used for the substitution of the incoming state bytes.

**Rcon:**
This function returns a 4 byte value based on the predefined values:

**Transformation functions used during encryption and decryption:**

**Shift_row decr:**
The 16 bytes are first converted to a 4X4 matrix. Each row of the matrix is then separately circularly shifted through (row-1) times to the left.

**Inverse Sub byte:**
A corresponding inverse of the S box is used for this function. The search process is same as that in Sub byte.

Algorithm for AES encryption is as follows:

**Algorithm for AES Cryptography (Sender Side)**

The AES key expansion takes 16 bytes of hex data and processes it for 43 rounds to expand it into 176 bytes of key. The key scheduling scheme uses transformation functions, SubWord(), RotWord(), and Rcon().

1. function key_expansion(key)
   while i <= 32
      make groups of key’s elements(key_array), each group being 8 hex or 8 words long
   end
   i=4;
   for i<=43
      temp = key_array(i)
      if ( mod(i, 4) = 0)
         temp = ( SubWord ( RotWord(temp) ) xor Rcon(i) ) xor key_array(i-3)
      else
         temp= key_array(i) xor key_array(i-3)
      end
      key_array(i+1) = temp
   end

2. Data block is of 64 bits as in case of DES encryption and key used here is expanded key of 176 bytes.

\[
\text{state1} = \text{Add\_round\_key} (\text{key}(1:16), \text{data\_block})
\]

\[
\text{for } i=2 \text{ to } 10
\]

\[
\begin{align*}
\text{state2} &= \text{Sub\_Bytes}\left(\text{state1}\right) \\
\text{state3} &= \text{Shift\_Row}\left(\text{state2}\right) \\
\text{state4} &= \text{Mix\_Column}\left(\text{state3}\right) \\
\text{state1} &= \text{Add\_round\_key} (\text{key}(((i-1)\times16)+1 : (i\times16)), \text{state4})
\end{align*}
\]

\[
\text{state2} = \text{Sub\_bytes}\left(\text{state1}\right)
\]

\[
\text{state3} = \text{Shift\_Row}\left(\text{state2}\right)
\]

\[
\text{state1} = \text{Add\_round\_key} (\text{key}((i\times16)+1 : ((i+1)\times16)), \text{state3})
\]

key(160:176) //

encrypted_data_block = state1
The decryption process has a process structure familiar to that of encryption. The major change in the decryption process is the use of key in reverse order. The transformation function are applied as follows:

1. The Add_round_key() function is same as that used in encryption, changing the state of data block. It takes last 16 key bytes along with the data block as input.

2. Then, as in encryption, this changed state is processed in a loop till upto 17th key byte is utilized in Add_round_key(). The use of function is as, inverse_Shift_Row() -> inverse_Sub_Bytes() -> Add_round_key() -> inverse_Mix_Column().

3. The last round utilizing 1:16 bytes of the key is carried out without inverse_Mix_Column(), same as done in encryption process.

Algorithm for AES decryption is:

**Algorithm for AES Cryptography (Receiver Side)**

1. Data block is again of 64 bits and key is of 176 bytes.
   
   \[
   \text{state1} = \text{Add\_round\_key(key(161:176), encrypted\_data\_block)}
   \]
   
   For \( i = 10 \) with steps of -1 to 2
   
   \[
   \begin{align*}
   \text{state2} &= \text{inverse\_Shift\_Row(state1)} \\
   \text{state3} &= \text{inverse\_Sub\_Bytes(state2)} \\
   \text{state4} &= \text{Add\_round\_key(key((i-1)*16)+1 : (i*16)), state3} \\
   \text{state1} &= \text{inverse\_Mix\_Column(state4)}
   \end{align*}
   \]
   
   end
   
   \[
   \begin{align*}
   \text{state2} &= \text{inverse\_Shift\_Row(state1)} \\
   \text{state3} &= \text{inverse\_Sub\_bytes(state2)} \\
   \text{state1} &= \text{Add\_round\_key(key(1 : 16)), state3}
   \end{align*}
   \]

2.4. **RC4 Stream Cipher**

RC4 is a variable key size stream cipher with byte oriented operations. It was designed by Ron Rivest for RSA security. The algorithm bases its operation on use of random permutation.

Figure 9 shows the block diagram of RC4 [5] encryption and decryption procedures. It includes two main processes,

1. Key scheduling algorithm (KSA)
2. Pseudorandom generation algorithm (PRGA).

In KSA, a variable length key (Key) of 1 to 256 bytes is used to initialise a 256 byte state vector ‘M’. The pseudorandom generation algorithm (PRGA) makes use of each state of the previous matrix to output the ciphered data using bit-xoring. A byte ‘kop’ is generated.
from 'M' by selecting one of the 255 entries in a symmetric order. The method of key stream generation is depicted in the following pseudo code.

### Pseudo code for KSA and PRGA in RC4 Cryptography

**KSA (Key Scheduling Algorithm):**
```
for k = 0 to 255 do
    M[k] = k;
    T[k] = Key[k mod keylength];

j = 0;
for k = 0 to 255 do
    j = (j + M[k] + T[k]) mod 256;
    Swap (M[k], M[j]);
```

**PRGA (pseudorandom generation algorithm):**
```
k, j = 0;
while (true)
    k = (k + 1) mod 256;
    j = (j + M[k]) mod 256;
    Swap (M[k], M[j]);
    t = (M[k] + M[j]) mod 256;
    kop = M[t];
```

The byte 'kop' thus obtained is XORed with the next byte of plain text to obtain the cipher text.

### Algorithm for RC4 Cryptography (Sender Side)
1. Get the image of size m*n and convert it into a 1-D stream, 'im_1D'.
2. Perform bit by bit xoring of 'kop' value and the image byte.
   ```
   for i=1 to m*n
       Cipher (i) = bitxor (kop(i),im_1D(i));
   ```
3. Get the cipher stream.
4. Convert it back into image dimensions.

For decryption, XOR the value of 'kop' with next byte of cipher text as shown in figure 9.

### Algorithm for RC4 Cryptography (Receiver Side)
1. Obtain the cipher image.
2. Convert it into 1-D data stream, 'cipher_1D'.
3. Perform bit by bit xoring of 'kop' value with the data stream.
   ```
   for i=1 to m*n
       original(i)=bitxor (kop(i),cipher_1D(i));
   ```
4. Obtain the 'original' data stream. Change the dimensions of stream to that of an image, to get the decrypted image.

### 2.5. IDEA Cryptography

International Data Encryption Algorithm (IDEA) [4] was first described in 1991. It is a block cipher which was originally called improved PES (improved Proposed Encryption Standard). This cipher works on 64 bits of plain text and uses a 128 bit key. In case of
image encryption whole image is subdivided into blocks of 64 bits each. Unlike DES and AES, this technique avoids the use of substitution boxes and the associated table lookups. For encryption, the plain text of 64 bits is divided into four blocks of 16 bits each. Key of 128 bits is divided to form 52 subkeys of 16 bits each. The procedure to form 52 subkeys from 128 bit key is described as follows:

- 128 bit key is first divided to form 8 subkeys of 16 bit. These subkeys are used directly.
- 128 bit key is the circularly shifted to 25 positions and the result obtained is then divided to again form 8 subkeys of 16 bits each. This operation is repeated until 52 different subkeys are generated.

Each round uses 6 subkeys and there are total 8 rounds (6X8=48). Last 4 subkeys are used to gets the cipher text of 64 bits. IDEA uses three different algebraic groups [3] in each round. Figure 10 shows the detailed structure of single round of IDEA algorithm.

This depicts the $2^{16}$ modulo addition of two 16 bit integers.
This depicts multiplication modulo $2^{16}+1$ of two 16 bit integers.
This depicts the bitwise exclusive or of two 16 bit integers.
Figure 10. IDEA Encryption Algorithm
Algorithm for IDEA Cryptography (Sender Side)

1. Obtain the image and convert it into its binary form. Size of image remains same (m*n). Each round is applied on a data of 64 bits.
2. Select a 128 bit key.
3. A set of 52 subkeys (16 bits each) is formed by applying circular rotations and divisions as explained in above paragraph. Step to form first 12 subkeys is shown below:
   \[
   \text{subkey}(1,:) = \text{key}; \\
   \text{for } i=2 \text{ to } 13 \\
   \quad \text{subkey}(i,:) = \text{circular shift(subkey}(i-1,:),[0 \ -25]); \\
   \text{end} \\
   \text{for } i=1 \text{ to } 6 \\
   \quad Z(i,:,1) = \text{subkey}(1,16*i-15:16*i); \\
   \text{end} \\
   \text{for } i=7 \text{ to } 8 \\
   \quad Z(i-6,:,2) = \text{subkey}(1,16*i-15:16*i); \\
   \text{end}
   \]
4. Perform modulo addition and modulo multiplication as specified in the figure above. First step of one round is shown below. Similarly, other steps are included to complete one round.
   \[
   \text{for round=1 to 8} \\
   \quad \text{data}(1,:) = \text{mod(data}(1,:)*Z(1,:,round),2^{16}+1); \\
   \quad \text{data}(4,:) = \text{mod(data}(4,:)*Z(4,:,round),2^{16}+1); \\
   \quad \text{data}(2,:) = \text{mod(data}(2,:)+Z(2,:,round),2^{16}); \\
   \quad \text{data}(3,:) = \text{mod(data}(3,:)+Z(3,:,round),2^{16}); \\
   \text{end}
   \]
5. Cipher image is obtained after applying complete 8 rounds on each block of 64 bits data.

Decryption using IDEA is similar to encryption mechanism but 52 subkeys are used in the reverse order.

Algorithm for IDEA Cryptography (Receiver Side)

1. Obtain the cipher image and the key.
2. The inverse subkeys ‘inverse_mul’ used for modulo multiplication are calculated as:
   \[
   [g,c,d] = \text{greatest common divisor } (2^{16}+1, \text{key}); \\
   \text{if } d<0 \\
   \quad d=2^{16}+1+d; \\
   \text{else if } d = 0 \\
   \quad d = 2^{16}; \\
   \text{else} \\
   \quad d=d; \\
   \text{end} \\
   \text{inverse_mul}=d;
   \]
3. The inverse subkeys ‘inverse_add’ for modulo addition are calculated as:
   \[
   \text{if key=0} \\
   \quad \text{inverse_add}=2^{16}; \\
   \text{else} \\
   \quad \text{Inverse_add}=2^{16}\cdot \text{key}; \\
   \text{end}
   \]
4. The subkeys used for decryption are generated in the following fashion. First four subkey generation ‘Z_de’ is shown below:

\[ r1=1; \]
\[ Z_{de}(1,:,r1)=\text{inverse}\_mul(Z_{de}(1,:,10-r1)); \]
\[ Z_{de}(2,:,r1)=\text{inverse}\_add(Z_{de}(2,:,10-r1)); \]
\[ Z_{de}(3,:,r1)=2^{16}\cdot Z_{de}(3,:,10-r1); \]
\[ Z_{de}(4,:,r1)=\text{inverse}\_mul(Z_{de}(4,:,10-r1)); \]

5. Step 4 of encryption procedure is employed with subkeys ‘Z_de’ (instead of ‘Z’ as shown in encryption algorithm), to obtain the original data.

6. Obtain the original image.

2.6. Visual Cryptography

Visual cryptography [11-17] is an encryption technique which allows information (pictures, text, etc.) to be encrypted in such a way that decryption becomes a mechanical operation that does not require a computer. Moni Naor and Adi Shami developed it in 1994. They demonstrated a visual secret sharing scheme, where an image was broken up into \( p \) shares so that only somebody with all \( p \) shares could decrypt the image, while any \( p-1 \) shares revealed no information about the original image. Each share was printed on a separate transparent sheet, and decryption was performed by overlaying the shares. When all \( p \) shares were overlaid, the original image would appear.

There is a simple algorithm for binary (black and white) visual cryptography that generates two encrypted messages called share1 and share2 from an original message. The algorithm for the above mentioned scheme is shown below.

**Algorithm for Visual Cryptography (Sender Side)**

1. Obtain a binary image.
2. \([\text{row column}]=\text{size(image)};\)
   \[
   \text{for } i=1 \text{ to row} \\
   \quad \text{for } j=1 \text{ to column} \\
   \quad \quad \text{Share1}(i,j)=\text{randi([0,1])} \\
   \quad \text{end} \\
   \text{end}
   \]
3. \(\text{for } i=1 \text{ to row} \)
   \[
   \text{for } j=1 \text{ to column} \\
   \quad \text{if (image}(i,j)==1) \\
   \quad \quad \text{share2}(i,j)=\text{share1}(i,j); \\
   \quad \text{else} \\
   \quad \quad \text{share2}(i,j)=\text{not(share1}(i,j)); \\
   \quad \text{end} \\
   \text{end}
   \]
4. Transmit (share1,share2).

Receiver side only needs to combine all the shares obtained. This combination procedure is performed with the help of XOR operator.
Algorithm for Visual Cryptography (Receiver Side)

1. Obtain share1 and share2.
2. Original image=XOR(share1,share2).

2.7. Hierarchical Cryptography

Hierarchical visual cryptography [12] encodes the image in two levels. Initially the image is encrypted in two different meaningless shares called share1 and share2 like in visual cryptography. This is the first level of Hierarchical visual cryptography. In the second level, these two shares are encrypted independently. Share1 is encrypted to yield share11 and share12. Similarly, share2 is encrypted to yield share21 and share22. At the end of second level of HVC four shares are available. Any three shares are taken and combined to form keyshare. The keyshare along with the remaining share is transmitted. The algorithm for encryption process is described below:

Algorithm for Hierarchical Visual Cryptography (Sender Side)

1. Obtain a binary image.
2. \([row \text{ column}]=\text{size(image)};\)
   
   \[
   \begin{align*}
   &\text{for } i=1 \text{ to row} \\
   &\text{for } j=1 \text{ to column} \\
   &\text{Share1}(i,j)=\text{randi}([0,1])
   \end{align*}
   \]

3. \(\text{for } i=1 \text{ to row} \)
   
   \[
   \begin{align*}
   &\text{for } j=1 \text{ to column} \\
   &\text{Share11}(i,j)=\text{randi}([0,1])
   \end{align*}
   \]

4. \(\text{for } i=1 \text{ to row} \)
   
   \[
   \begin{align*}
   &\text{for } j=1 \text{ to column} \\
   &\text{Share21}(i,j)=\text{randi}([0,1])
   \end{align*}
   \]

5. \(\text{for } i=1 \text{ to row} \)
   
   \[
   \begin{align*}
   &\text{for } j=1 \text{ to column} \\
   &\text{if } (\text{image}(i,j)==1) \\
   &\text{share } 2(i,j)=\text{share1}(i,j); \\
   &\text{else} \\
   &\text{share2}(i,j)=\text{not(share1}(i,j));
   \end{align*}
   \]

6. \(\text{for } i=1 \text{ to row} \)
for j=1 to column
    if (share1(i,j)==1)
        share12(i,j)=share11(i,j);
    else
        share12(i,j)=not(share11(i,j));
    end
end
end
7. for i=1 to row
    for j=1 to column
        if (share2(i,j)==1)
            share22(i,j)=share21(i,j);
        else
            share22(i,j)=not(share21(i,j));
        end
    end
end
8. keyshare=XOR(share12(i,j),XOR(share21(i,j),share22(i,j))).
9. Transmit (keyshare,share11).

Receiver side only needs to combine all the shares obtained. This combination procedure is performed with the help of XOR operator.

Algorithm for Hierarchical Visual Cryptography (Receiver Side)

1. Obtain keyshare and share11.
2. Original image=XOR(keyshare,share11).

2.8. Elliptic Curve Cryptography

Elliptic curve cryptography [8, 15, 16] is the public key mechanism. It has many advantages over other public key cryptosystems such as RSA. In this technique, an elliptic curve ‘E’ (y = x^2 + a*x + b) is assumed.

A point ‘P’ is selected on the curve and is used as a part of public key and private key is taken to be any random integer ‘k’. Q = k*P is calculated and E, P, n and Q together makes the complete public key. Encryption procedure is described as follows:

Algorithm for Elliptic Curve Cryptography (Sender Side)

1. Obtain the image to be encrypted, convert all pixels into their binary equivalents and store them all in a 1-D array. Size of image is m*n (m rows and n columns) and size of 1-D array becomes, m*n*8.
2. Embed the binary values in the curve ‘E’ to form ‘Pm’.
   for i=1 to m*n*8
       x=\text{array}(i);
       y=x^2+a*x+b;  \quad \text{*curve E, a and b are random values*}
       Pm=[x,y];
   end
3. A random integer, ‘d’ is selected such that d < n.
\[ d = \text{randi}([1,n-1]); \]

4. Select a point random point \( P \) on curve \( E \). Calculate \( Q = d*P \).

5. \( P_m + d*Q \) and \( d*P \) are calculated.
   \[ C1 = d*P; \]
   \[ C2 = P_m + d*Q; \]

6. \((d*P, P_m+d*Q)\) is the encrypted pixel. It is converted back into the matrix and sent it as an encrypted image.

\[
\begin{align*}
\text{v1} &= \text{de2bi}(C1); \\
\text{v2} &= \text{de2bi}(C2); \\
[m1 \ n1] &= \text{size}(\text{v1}); \\
[m2 \ n2] &= \text{size}(\text{v2}); \\
j &= 1; \ e &= 1; \\
\text{for } i = 1 \text{ to } 2*m1 \\
& \quad \text{cipher1}(e:e+n1-1) = \text{v1}(j,:); \\
& \quad e = e + n1; \\
& \quad j = j + 1; \\
& \quad \text{if } i == m1 \\
& \quad \quad \text{v1} = \text{v2}; \\
& \quad \quad j = 1; \\
& \quad \quad n1 = n2; \\
& \text{end}
\end{align*}
\]

Decryption procedure is as follows:

Algorithm for Elliptic Curve Cryptography (Receiver Side)

1. Obtain the encrypted image and convert it into a 1-D array (encrypt).
2. Decryption is done by using the private key ‘\( k \).
3. Message \( (m) \) can be decoded as, \( m = (P_m+d*Q) - d*k*P \).
   \[
   \text{for } i = 1 \text{ to length(encrypt)}
   \]
   \[ m(i) = C2(i,1) - k*C1(i,1); \]
   \[ \text{end} \]
4. This is again converted into matrix form to get the original image.

2.9. Vigenère Cryptography

It is a multi-alphabetic substitution encryption which uses two or more cipher alphabets. First step in this encoding process is to form a vigenère \([10, 18]\) table which is as follows. In this table each row is formed by shifting circularly the sequence of characters towards left. First row is formed by 0 alphabets shifting, second row by one step shifting and third row by two steps shifting and like this whole table can be formed. Key is of fixed size and value. Find the letter of key on row side and data to be encrypted on column side. For each particular value of row and column, an alphabet is present and this alphabet is used as the encrypted letter. Decryption of data is also done using same symmetric table. But as images do not use alphabets, this technique has been modified to work on numbers as each pixel of an image consists of an 8 bit value. The mathematical interpretation of modified vigenère cipher is as follows:

Encryption
\( \text{ciphered pixel} = \text{mod}(\text{pixel}+\text{key}),256) \)

Decryption
\( \text{original message} = \text{mod}(\text{ciphered pixel-\text{key}},256) \)
The key must be cycled until all the pixels are encrypted (like for short keys, if K is the key, KK, KKK and so on can be the equivalent keys). For example: suppose we have a pixel value 10, and the hex key as ’0B’, this hex value corresponds to a decimal value of 11, the encryption process will be carried out as:

ciphered pixel = mod((10+11), 256) = 21

and the decryption process will be as follows:

original message = mod((21-11), 256) = 10.

Detailed description of the encryption algorithm is depicted below:

Algorithm for Vigenère Cryptography (Sender Side)

1. Obtain the image. Convert it into gray scale. Suppose image comprises of ‘m’ rows and ‘n’ columns.
2. Obtain a key of 32 hex values. Convert it into a 1-D array of decimal values.
   
   \[
   \text{for } i = 1 \text{ to length(key)} \\
   \quad \text{key_str} = \text{char(key}(i)\text{)}; \\
   \quad \text{key_dec}(i) = \text{hex2dec(key_str)}; \\
   \text{end}
   \]
3. Encrypted image ‘encrypt’ can be formed by following steps below:
   
   \[
   \text{for } i = 1 \text{ to m} \\
   \quad \text{pntr=1}; \\
   \quad \text{while } \text{pntr} < n \\
   \quad \quad \text{for } j = 1 \text{ to size(key)} \\
   \quad \quad \quad \text{encrypt}(i, \text{pntr}) = \text{mod(key}(1,j)\text{)+image}(i, \text{pntr}), 256); \\
   \quad \quad \quad \text{pntr} = \text{pntr} + 1; \\
   \quad \quad \text{end} \\
   \quad \text{end}
   \]

Description for decryption using vigenère scheme is given below:

Algorithm for Vigenère Cryptography (Receiver Side)

1. Obtain the encrypted image and the key.
2. Get the decrypted or original image ‘decrypt’ by applying the following procedure:
   
   \[
   \text{for } i = 1 \text{to m} \\
   \quad \text{pntr}=1; \\
   \quad \text{while } \text{pntr} < n \\
   \quad \quad \text{for } j = 1 \text{to size(key)} \\
   \quad \quad \quad \text{decrypt}(i, \text{pntr}) = \text{mod(encrypt}(i, \text{pntr})\text{-key}(1,j), 256); \\
   \quad \quad \quad \text{pntr} = \text{pntr} + 1; \\
   \quad \quad \text{end} \\
   \quad \text{end}
   \]

2.10. Chaotic Encryption Technique

This encryption scheme is based on a Peter De Jong Chaotic Map [25] and RC4 Stream Cipher. This technique is comprised of a round key generation function and three encryption functions as shown in figure 11. Original image is converted to grey image and
stored in ‘I’ whose size is M x N. 128 Peter De Jong Chaotic [21, 22] values are generated and converted into integer form. An initial set of parameters (X0, Y0, a, b, c, d) is taken. The (X, Y) values becomes the initial key for RC4 stream generator. To increase the security this stream generation and encryption can be carried out multiple times.

1. Permutation: M + N Peter De Jong Chaotic values are generated. X1 to XM values are stored in array A while Y1 to YN in array B. Both A and B are sorted and the positions of sorted chaotic [23, 24] values in original sequence are found and stored in array C and D. Each and every row is circularly rotated in alternate orientations and amount of rotation is based on array C. Circular rotation of each column in alternate orientation is done based on array D.

2. Pixel Value Rotation: M x N RC4 pseudo random numbers are generated and mod 8 is applied to obtain all values in the range of 1 to 7. These numbers are used for circular rotations by first reading the image horizontally and applying alternate left/right circular rotation on pixel to pixel basis and then, by reading it vertically and applying alternate left/right circular rotation on pixel by pixel basis.

3. Diffusion: Image is scanned row wise and then column wise in alternate orientations and the forward and backward diffusion is applied.

**Algorithm for Chaotic Cryptography (Sender Side)**

Key set = \{X0; Y0; a; b; c; d\}

**Generating Keyset**

1. Generate 128 Peter De Jong chaotic values \{X and Y sequence\} using keyset & Transform it to the integer form.

2. Assign 128 X sequence values to Key[0], ..., Key[127], and Y sequence to Key[128], ..., Key[255]

---

**Figure 11. Block Diagram of Chaotic Encryption**

The detailed description of chaotic encryption algorithm is depicted below:
**Permutation**

1. M & N Peter De Jong chaotic values \( X_1 = (X_1, \ldots, X_M), Y_1 = (Y_1, \ldots, Y_N) \) are generated. (M and N are rows and columns of image).

2. Sort \( X_1 \), and \( Y_1 \), Lets call the positions of the sorted chaotic values in the original chaotic sequence are found and are stored in \( X_2, Y_2 \).

   \[
   for \ i=1:M \\
   \quad X2(i)=\text{find}(X1=\text{value at loc i of sorted array}); \\
   \text{end} \\
   \text{Similarly, for Columns same approach is applied}
   \]

3. Scramble rows and columns according to \( X_2 \), and \( Y_2 \) values respectively. eg swap the row values of \( X_2(1) \) row with \( X_2(2) \) and so on.

   \[
   for \ i=1:M \\
   \quad \text{swap}(\text{Image (i,:) = Image}(X2(i,:))); \\
   \text{end} \\
   \text{Similarly, for Columns same approach is applied}
   \]

4. Circularly rotate each and every row in alternate orientations and the amount of rotation is based on \( X_2 \). eg. The first row is rotated by \( X_2(1) \), the second row is rotated by \( X_2(2) \), and so on (- means opposite direction)

   \[
   for \ i=1\text{to } M \\
   \quad \text{if(odd value of i)\% alternate orientation} \\
   \quad \quad \text{Image(i,:) = circshift(Image(i,:), -X2(i));} \\
   \quad \text{else} \\
   \quad \quad \text{Image(i,:) = circshift(Image(i,:), X2(i));} \\
   \quad \text{end} \\
   \text{end} \\
   \text{Similarly, for Columns same approach is applied}
   \]

5. Repeat this step with columns, using \( Y_2 \) values.

**Pixel Value Circular Rotations**

M * N RC4 pseudo random numbers determine the amount of rotation for the circular rotation,

1. Generate M * N RC4 pseudo random numbers

2. Read the image horizontally and then apply alternate left/right Circular rotation on pixel by pixel basis. Repeat the same in vertical direction

   \[
   for \ i=1\text{to } M \\
   \quad \text{for } j=1\text{to } N \\
   \quad \text{if(odd value of i)\% alternate orientation} \\
   \quad \quad \text{Image(i,j)= pixelshift(Image(i,j),-Psudo_random(i));} \\
   \quad \text{else} \\
   \quad \quad \text{Image(i,j)= pixelshift(Image(i,j), Psudo_random(i));} \\
   \quad \text{end}
   \]
Similarly, for Vertical direction same approach can be applied

** Diffusion

1. forward and backward diffusion is applied after scanning the image row-wise in alternate orientations.

2. Here image is scanned column wise in alternate orientation and the farward and backward diffusion is applied (I = Image scanned row wise)

Forward Diffusion:

\[
E_i = (((((P_i + E_i - 2) \mod 256) + E_i - 1) \mod 256 \xor R_1i) + R_2i) \mod 256; i = 1; 2; \ldots; MN
\]

Backward Diffusion:

\[
E_i = (((((E_i + F_i + 2) \mod 256) + F_i + 1) \mod 256 \xor R_1i) + R_2i) \mod 256; i = MN; \ldots; 1
\]

% % forward diffusion
for i = 1 to M*N
% % Permuted and rotated pixel
P = I(i);
% % foremely encrypted pixels
E2 = Diffused_row_far(i-2);
E1 = Diffused_row_far(i-1);
end
% % Pseudo random no.
R1_num = R1(i);
R2_num = R2(i);
% % forward diffusion
Diffused_row_far(i)= forward_diffusion(P,E2,E1,R1_num,R2_num);
end
% % backward diffusion
for i = M*N to 1
% % Permuted and rotated pixel
E = Diffused_row_far (i);
% % foremely encrypted pixels
F2 = Diffused_row_back(i+2);
F1 = Diffused_row_back(i+1);
end
% % Pseudo random no.
R1_num = R1(i);
R2_num = R2(i);
% % forward diffusion
Diffused_row_back(i)= backward_diffusion(E,F2,F1,R1_num,R2_num);
end
Similarly, for Column direction same approach can be applied.

Decryption includes reconstruction of the original image by following inverse steps. First of all, diffusion is applied on the encrypted image then pixel values are circularly rotated in reverse direction and finally the inverse permutation is applied to obtain the original image.

3. Simulation Setup Parameters

Setup parameters are given in table 2.

<table>
<thead>
<tr>
<th>Table 2. Setup Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Image size</strong></td>
</tr>
<tr>
<td><strong>Image type</strong></td>
</tr>
<tr>
<td><strong>Simulation tool</strong></td>
</tr>
<tr>
<td><strong>Processor</strong></td>
</tr>
<tr>
<td><strong>Cryptographic Parameters</strong> (AES)</td>
</tr>
<tr>
<td><strong>Cryptographic Parameters</strong> (DES)</td>
</tr>
<tr>
<td><strong>Cryptographic Parameters</strong> (3-DES)</td>
</tr>
<tr>
<td><strong>Vigenère Cryptography</strong></td>
</tr>
<tr>
<td><strong>IDEA Cryptography</strong></td>
</tr>
<tr>
<td><strong>Visual Cryptography</strong></td>
</tr>
<tr>
<td><strong>Hierarchical Cryptography</strong></td>
</tr>
<tr>
<td><strong>RC4 Cryptography</strong></td>
</tr>
<tr>
<td><strong>Elliptic Curve Cryptography</strong></td>
</tr>
<tr>
<td><strong>Chaotic Cryptography</strong></td>
</tr>
</tbody>
</table>

4. Performance Metrics

For complete analysis of the techniques, numerous parameters have been used. This section deals with the details of all those parameters.
4.1. Visual Assessment

Good encryption technique means there should be no visual information in the encrypted images. Encrypted images should be random-like and highly disordered. Any intruder should not be able to interpret anything from the encrypted image.

4.2. Key-Space Analysis

Image encryption schemes should be sensitive to keys or any initial parameters which are used during encryption. Large key space diminishes brute force attacks. Large is the key space, more is the time taken to decode the key.

4.3. Statistical Analysis

This analysis is done to analyse the confusion and diffusion properties of an encrypted image. Evaluation of correlation coefficients signifies how strongly the pixels are related to each other and histograms demonstrate pixel relation in frequency domain. Correlation analysis and histogram analysis can specify which technique has better confusion and diffusion properties, which can resist statistical attacks.

4.4. Histogram Analysis

Histograms of an image is the frequency description of each pixel value. Histograms of encrypted images should be completely statistically different from original images. For the technique to be resistant to statistical histogram attacks, encrypted images should have uniform histograms.

4.5. Correlation Analysis

This parameter calculates the correlation among the adjacent pixels of an image. A good encryption technique should result into an encrypted image with no or zero correlation between adjacent pixels. In order to test the correlation between two adjacent pixels in plain-image and cipher-image, we have analysed the correlation between pairs of plain-image channels and cipher-image channels. The following formulae is used to calculate the correlation coefficients in the horizontal, vertical and diagonal directions.

\[
\begin{align*}
ra\beta &= \frac{\text{cov}(\alpha, \beta)}{\sqrt{\text{D}(\alpha)} \sqrt{\text{D}(\beta)}} \\
E(\alpha) &= \frac{1}{N} \sum_{i=1}^{N} \alpha_i \\
E(\alpha) &= \frac{1}{N} \sum_{i=1}^{N} (\alpha_i - E(\alpha))^2 \\
\text{cov}(\alpha, \beta) &= \frac{1}{N} \sum_{i=1}^{N} (\alpha_i - E(\alpha))(\beta_i - E(\beta))
\end{align*}
\]

where \(\alpha\) and \(\beta\) denote two adjacent pixels and \(N\) is the total number of duplets \((\alpha, \beta)\) obtained from the image.

4.6. Differential analysis

An image encryption technique should be sensitive to a small change in key and plain image. In order to assess the changing a single bit of key or any pixel value in the plain-image on the cipher-image, the number of pixel change rate (NPCR) and the unified averaged changing intensity (UACI) are computed in the encryption scheme. The NPCR is used to measure the change rate of the number of pixels of the cipher-image when only one bit of key or pixel is modified. The UACI measures the average intensity of two one bit changes of cipher-images. Let us assume, the two ciphered images \(C_1\) and \(C_2\) whose
corresponding plain images have only one-pixel difference. The color RGB-level values of ciphered images \( C_1 \) and \( C_2 \) at row \( i \), column \( j \) are labelled as \( C_1(i,j) \) and \( C_2(i,j) \), respectively. The NPCR is defined as

\[
NPCR = \frac{\sum_{i,j} D(i,j)}{MN} \times 100\%
\]

where \( M \) and \( N \) are the width and height of two random images and \( D(i,j) \) is defined as

\[
D(i,j) = f(x) = \begin{cases} 
1, & \text{if } C_1(i,j) \neq C_2(i,j) \\
0, & \text{otherwise}
\end{cases}
\]

Further, the UACI, is used to measure the average intensity difference in a color component between the two cipher images \( C_1(i,j) \) and \( C_2(i,j) \). It is defined as

\[
UACI = \frac{1}{MN} \left[ \sum_{i,j} \frac{|C_1(i,j)-C_2(i,j)|}{255} \right] \times 100\%
\]

4.7. Key Sensitivity

Key sensitivity means that a slightly different key produces an entirely different cipher image. NPCR and UACI are calculated to measure difference between two cipher images. One being encrypted using the original key and other one with a single bit change in the key. NPCR > 99% and UACI around 33% shows that the scheme can resist differential attack.

4.8. Pixel/Plain image sensitivity

Plain image sensitivity infers that even a single bit change in the plain image should produce a completely different encrypted image. NPCR and UACI are calculated to check the difference between the two encrypted images. One bit is altered in the plain image and then it is encrypted using the same key and initial parameters to form the second cipher image, first being the cipher formed from the original image. The scheme showing NPCR over 99% and UACI over 33% is considered to be secure against differential attack and to be highly sensitive to pixel change.

4.9. Information Entropy Analysis

Information entropy is the amount of randomness in information content of the image. The entropy \( H(S) \) of a message \( m \) can be calculated as

\[
H(S) = \sum_{i=0}^{n-1} P(S_i) \log_2 \frac{1}{P(S_i)}
\]

where \( P(S_i) \) represents the probability of the occurrence of symbol \( S_i \) and \( \log \) denotes the base 2 logarithm. If there are 256 possible outcomes of the message \( S \) with equal probability, it is considered as random. In this case, \( H(S) = 8 \), is an ideal value. The value of entropy close to value 8 signifies that the encryption algorithm is secure against entropy attack.

4.10 Peak Signal to Noise Ratio (PSNR) analysis

In PSNR analysis, the original image is considered as the signal while the encrypted image is taken as the noise. The MSE stands for cumulative squared error between the stego image and the original image. Lower the value of MSE means lower error. It is defined by the relation given below any \( m \times n \) monochrome image. Where, \( MAX_i \) represents maximum value of pixel of the image. In the images with pixel having 8 bits per sample, its value is 255.
Formula used for calculation of PSNR are given below.

\[ MSE = \frac{1}{mn} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} [I(i, j) - K(i, j)]^2 \]  
(9)

\[ PSNR = 10 \log_{10}(\frac{MAX^2}{MSE}) \]  
(10)

4.11 Computational Speed Analysis

As images are bulky, so the encryption mechanism should be fast and take less encryption time. Computational speed may depend upon the type of processor used, type of programming language and type of encryption technique.

5. Results

5.1. Visual Assessment

Results of various techniques have been given in table 3.

<table>
<thead>
<tr>
<th>Encryption Technique</th>
<th>Plain image</th>
<th>Encrypted image</th>
<th>Decrypted image</th>
</tr>
</thead>
<tbody>
<tr>
<td>DES</td>
<td>![Image]</td>
<td>![Image]</td>
<td>![Image]</td>
</tr>
<tr>
<td>3-DES</td>
<td>![Image]</td>
<td>![Image]</td>
<td>![Image]</td>
</tr>
<tr>
<td>AES</td>
<td>![Image]</td>
<td>![Image]</td>
<td>![Image]</td>
</tr>
<tr>
<td>Elliptic</td>
<td>![Image]</td>
<td>![Image]</td>
<td>![Image]</td>
</tr>
<tr>
<td>Idea</td>
<td>![Image]</td>
<td>![Image]</td>
<td>![Image]</td>
</tr>
</tbody>
</table>
As it can be seen from Table 3, encrypted output of the encryption schemes is totally different from original data and no meaningful information can be deduced from it.

5.2. Key Space Analysis

Table 4 gives the comparison of key space of all the techniques implemented.

<table>
<thead>
<tr>
<th>Cipher</th>
<th>Key length</th>
<th>Key space</th>
</tr>
</thead>
<tbody>
<tr>
<td>DES</td>
<td>56</td>
<td>$2^{56}$</td>
</tr>
<tr>
<td>3DES</td>
<td>168</td>
<td>$2^{168}$</td>
</tr>
<tr>
<td>AES</td>
<td>128</td>
<td>$2^{128}$</td>
</tr>
<tr>
<td>RC4</td>
<td>256</td>
<td>$2^{256}$</td>
</tr>
<tr>
<td>Elliptic</td>
<td>$2^64$</td>
<td>$2^{128}$</td>
</tr>
<tr>
<td>Idea</td>
<td>128</td>
<td>$2^{128}$</td>
</tr>
<tr>
<td>Vigenère</td>
<td>128</td>
<td>$2^{128}$</td>
</tr>
<tr>
<td>Chaotic</td>
<td>$6^64$</td>
<td>$2^{384}$</td>
</tr>
</tbody>
</table>

Out of all the schemes implemented it is clearly visible that chaotic scheme has the largest key space that is $2^{384}$. 
5.3. Statistical Analysis

To avoid statistical attacks histogram analysis and correlation analysis should be done. The results of these both analysis have been shown in sections below.

5.3.1. Histogram Analysis:

<table>
<thead>
<tr>
<th>Encryption Technique</th>
<th>Plain Image</th>
<th>Encrypted Image</th>
<th>Decrypted Image</th>
</tr>
</thead>
<tbody>
<tr>
<td>DES</td>
<td><img src="image1" alt="Graph" /></td>
<td><img src="image2" alt="Graph" /></td>
<td><img src="image3" alt="Graph" /></td>
</tr>
<tr>
<td>3-DES</td>
<td><img src="image4" alt="Graph" /></td>
<td><img src="image5" alt="Graph" /></td>
<td><img src="image6" alt="Graph" /></td>
</tr>
<tr>
<td>AES</td>
<td><img src="image7" alt="Graph" /></td>
<td><img src="image8" alt="Graph" /></td>
<td><img src="image9" alt="Graph" /></td>
</tr>
<tr>
<td>Elliptic</td>
<td><img src="image10" alt="Graph" /></td>
<td><img src="image11" alt="Graph" /></td>
<td><img src="image12" alt="Graph" /></td>
</tr>
<tr>
<td>Idea</td>
<td><img src="image13" alt="Graph" /></td>
<td><img src="image14" alt="Graph" /></td>
<td><img src="image15" alt="Graph" /></td>
</tr>
<tr>
<td>RC4</td>
<td><img src="image16" alt="Graph" /></td>
<td><img src="image17" alt="Graph" /></td>
<td><img src="image18" alt="Graph" /></td>
</tr>
<tr>
<td>Visual</td>
<td><img src="image19" alt="Graph" /></td>
<td><img src="image20" alt="Graph" /></td>
<td><img src="image21" alt="Graph" /></td>
</tr>
</tbody>
</table>
Hierarchical

Table 5 completely depicts the differences in the histograms of encrypted images obtained after applying different cryptography mechanisms. Histogram of the chaotic technique proves to be best amongst all as it is most uniform and avoids to leak any information to the intruder by just analysing the histogram of the encrypted image.

5.3.2. Correlation Analysis

Table 6. Correlation Analysis (128*128)

<table>
<thead>
<tr>
<th></th>
<th>Horizontal cor</th>
<th>Vertical cor</th>
<th>Diagonal cor</th>
</tr>
</thead>
<tbody>
<tr>
<td>128x128</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DES</td>
<td>0.28075</td>
<td>-0.04335</td>
<td>-0.01436</td>
</tr>
<tr>
<td>3DES</td>
<td>0.256303</td>
<td>0.016179</td>
<td>0.013917</td>
</tr>
<tr>
<td>AES</td>
<td>0.192696</td>
<td>-0.00748</td>
<td>-0.00302</td>
</tr>
<tr>
<td>RC4</td>
<td>0.007782</td>
<td>0.005397</td>
<td>0.00822</td>
</tr>
<tr>
<td>Elliptical</td>
<td>0.226187</td>
<td>-0.29583</td>
<td>-0.08822</td>
</tr>
<tr>
<td>Visual</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>share1</td>
<td>0.025931</td>
<td>0.001132</td>
<td>-0.01622</td>
</tr>
<tr>
<td>share2</td>
<td>0.005039</td>
<td>-0.01608</td>
<td>0.002604</td>
</tr>
<tr>
<td>Hierarchical</td>
<td>0.009229</td>
<td>0.009746</td>
<td>-0.02127</td>
</tr>
<tr>
<td>share1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>share2</td>
<td>-0.00021</td>
<td>0.001722</td>
<td>0.002928</td>
</tr>
<tr>
<td>Idea</td>
<td>0.626377</td>
<td>-0.07712</td>
<td>-0.05115</td>
</tr>
<tr>
<td>Vigenère</td>
<td>0.920082</td>
<td>-0.08558</td>
<td>-0.09983</td>
</tr>
<tr>
<td>Chaotic</td>
<td>-0.00084</td>
<td>0.000143</td>
<td>-0.00085</td>
</tr>
</tbody>
</table>

Table 7. Correlation Analysis (256*256)

<table>
<thead>
<tr>
<th></th>
<th>Horizontal cor</th>
<th>Vertical cor</th>
<th>Diagonal cor</th>
</tr>
</thead>
<tbody>
<tr>
<td>256x256</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DES</td>
<td>0.223646</td>
<td>-0.0072</td>
<td>0.008191</td>
</tr>
<tr>
<td>3DES</td>
<td>0.025587</td>
<td>0.02113</td>
<td>0.005366</td>
</tr>
<tr>
<td>AES</td>
<td>-0.00644</td>
<td>0.004249</td>
<td>-0.00816</td>
</tr>
<tr>
<td>RC4</td>
<td>-0.01506</td>
<td>-0.00355</td>
<td>-0.00572</td>
</tr>
<tr>
<td>Elliptical</td>
<td>-0.36302</td>
<td>-0.37163</td>
<td>0.39189</td>
</tr>
</tbody>
</table>
Correlation analysis has been shown by the above three tables for different image sizes. Table 6 shows correlation analysis for size 128*128, table 7 for size 256*256 while table 8 shows results for size 384*384. All the three tables depict that hierarchical cryptography, visual cryptography and RC4 stream cipher have less correlation among adjacent pixels but chaotic cryptography has the least amongst all. So according to this analysis, chaotic cryptography emerges as the best one.

### 5.4. Sensitivity Analysis

This section includes results for key and pixel sensitivity on three different size of images.

#### 5.4.1. Key Sensitivity Analysis

<table>
<thead>
<tr>
<th>Key Sensitivity</th>
<th>NPCR</th>
<th>128x128</th>
<th>256x256</th>
<th>384x384</th>
</tr>
</thead>
<tbody>
<tr>
<td>DES</td>
<td>99.78637</td>
<td>99.7039</td>
<td>99.59649</td>
<td></td>
</tr>
<tr>
<td>3DES</td>
<td>99.5788</td>
<td>99.54681</td>
<td>99.61954</td>
<td></td>
</tr>
<tr>
<td>AES</td>
<td>99.60327</td>
<td>99.59106</td>
<td>99.60734</td>
<td></td>
</tr>
<tr>
<td>RC4</td>
<td>99.70703</td>
<td>99.59869</td>
<td>99.59581</td>
<td></td>
</tr>
<tr>
<td>Elliptical</td>
<td>94.37255</td>
<td>95.05463</td>
<td>94.94154</td>
<td></td>
</tr>
</tbody>
</table>
Table 10. UACI Analysis for Modified Key

<table>
<thead>
<tr>
<th>key sensitivity</th>
<th>UACI</th>
<th>128x128</th>
<th>256x256</th>
<th>384x384</th>
</tr>
</thead>
<tbody>
<tr>
<td>Visual share1</td>
<td>95.12939</td>
<td>95.34149</td>
<td>95.29012</td>
<td></td>
</tr>
<tr>
<td>share2</td>
<td>85.5951</td>
<td>85.31800</td>
<td>85.69539</td>
<td></td>
</tr>
<tr>
<td>Hierarchical</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>share1</td>
<td>51.1111</td>
<td>51.42212</td>
<td>50.11868</td>
<td></td>
</tr>
<tr>
<td>share2</td>
<td>60.88867</td>
<td>60.40192</td>
<td>46.70817</td>
<td></td>
</tr>
<tr>
<td>Idea</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>share1</td>
<td>99.68261</td>
<td>99.60327</td>
<td>99.6154785</td>
<td></td>
</tr>
<tr>
<td>share2</td>
<td>33.43711</td>
<td>33.729505</td>
<td>33.469275</td>
<td></td>
</tr>
</tbody>
</table>

Table 9 and table 10 shows the comparison of NPCR and UACI for a slight change in the initial key used. Chaotic, DES and RC4 have comparable results for NPCR and prove to be better than others in the table 9. Table 10 depicts the chaotic to be best for highest UACI amongst all.

5.4.2. Pixel Sensitivity analysis

Table 11. NPCR Analysis for Modified Pixel

<table>
<thead>
<tr>
<th>plain image sensitivity</th>
<th>NPCR</th>
<th>128x128</th>
<th>256x256</th>
<th>384x384</th>
</tr>
</thead>
<tbody>
<tr>
<td>DES</td>
<td>0.048828</td>
<td>0.012207</td>
<td>0.005425</td>
<td></td>
</tr>
<tr>
<td>3DES</td>
<td>0.048828</td>
<td>0.012207</td>
<td>0.005425</td>
<td></td>
</tr>
<tr>
<td>AES</td>
<td>0.097656</td>
<td>0.024414</td>
<td>0.010851</td>
<td></td>
</tr>
<tr>
<td>RC4</td>
<td>99.59106</td>
<td>99.57122</td>
<td>99.63718</td>
<td></td>
</tr>
<tr>
<td>Elliptical</td>
<td>98.71728</td>
<td>96.66675</td>
<td>96.6132</td>
<td></td>
</tr>
<tr>
<td>Visual share1</td>
<td>85.98022</td>
<td>85.50111</td>
<td>85.36377</td>
<td></td>
</tr>
<tr>
<td>share2</td>
<td>85.74829</td>
<td>85.48126</td>
<td>85.28849</td>
<td></td>
</tr>
<tr>
<td>Hierarchical share1</td>
<td>95.15381</td>
<td>95.31708</td>
<td>95.22434</td>
<td></td>
</tr>
<tr>
<td>share2</td>
<td>85.5896</td>
<td>85.66589</td>
<td>85.56857</td>
<td></td>
</tr>
<tr>
<td>Idea</td>
<td>0.1892</td>
<td>0.047302</td>
<td>0.021021</td>
<td></td>
</tr>
</tbody>
</table>
Table 12. UACI Analysis for Modified Pixel

<table>
<thead>
<tr>
<th>plain image sensitivity</th>
<th>UACI 128x128</th>
<th>256x256</th>
<th>384x384</th>
</tr>
</thead>
<tbody>
<tr>
<td>DES</td>
<td>0.009885301</td>
<td>0.0037997</td>
<td>0.001183</td>
</tr>
<tr>
<td>3DES</td>
<td>0.016252106</td>
<td>0.0057145</td>
<td>0.001293</td>
</tr>
<tr>
<td>AES</td>
<td>0.03784179</td>
<td>0.0100408</td>
<td>0.003856</td>
</tr>
<tr>
<td>RC4</td>
<td>33.7229</td>
<td>33.52</td>
<td>33.6581</td>
</tr>
<tr>
<td>Elliptical</td>
<td>29.65049</td>
<td>33.4245</td>
<td>36.16463</td>
</tr>
<tr>
<td>Visual</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>share1</td>
<td>49.82833</td>
<td>49.75162</td>
<td>49.49279</td>
</tr>
<tr>
<td>share2</td>
<td>49.83431</td>
<td>49.74495</td>
<td>49.48651</td>
</tr>
<tr>
<td>Hierarchical</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>share1</td>
<td>48.2023351</td>
<td>48.3676147</td>
<td>48.2556</td>
</tr>
<tr>
<td>share2</td>
<td>49.5082481</td>
<td>49.64775</td>
<td>49.56005</td>
</tr>
<tr>
<td>Idea</td>
<td>0.00074199</td>
<td>0.000185499</td>
<td>8.24E-05</td>
</tr>
<tr>
<td>Vigenère</td>
<td>2.39354E-05</td>
<td>5.98384E-06</td>
<td>2.66E-06</td>
</tr>
<tr>
<td>chaotic</td>
<td>33.25583</td>
<td>33.5760258</td>
<td>33.178189</td>
</tr>
</tbody>
</table>

Table 11 and 12 shows the comparison of NPCR and UACI for a single bit change in the original image. NPCR is best for chaotic cryptography as shown by table 11. Chaotic cryptography proves to be best for UACI for a single bit change in the original image.

5.5. Entropy Analysis

Table 13. Entropy Analysis

<table>
<thead>
<tr>
<th>Entropy</th>
<th>128x128</th>
<th>256x256</th>
<th>384x384</th>
</tr>
</thead>
<tbody>
<tr>
<td>DES</td>
<td>7.87846</td>
<td>7.992748</td>
<td>7.998581</td>
</tr>
<tr>
<td>3DES</td>
<td>7.87729</td>
<td>7.993408</td>
<td>7.998705</td>
</tr>
<tr>
<td>AES</td>
<td>7.968921</td>
<td>7.997115</td>
<td>7.998785</td>
</tr>
<tr>
<td>RC4</td>
<td>7.98768</td>
<td>7.99584</td>
<td>7.996573</td>
</tr>
<tr>
<td>Elliptical</td>
<td>6.540688</td>
<td>6.674694</td>
<td>7.104149</td>
</tr>
<tr>
<td>Visual</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>share1</td>
<td>4.077702</td>
<td>4.032366</td>
<td>4.027419</td>
</tr>
<tr>
<td>share2</td>
<td>4.059462</td>
<td>4.032138</td>
<td>4.030686</td>
</tr>
<tr>
<td>Hierarchical</td>
<td>5.512987</td>
<td>5.496993</td>
<td>5.511118</td>
</tr>
<tr>
<td>share1</td>
<td>4.032961</td>
<td>4.013899</td>
<td>4.041041</td>
</tr>
<tr>
<td>share2</td>
<td>0.99999</td>
<td>0.999984</td>
<td>0.999973</td>
</tr>
<tr>
<td>Idea</td>
<td>7.693441</td>
<td>7.673212</td>
<td>7.979659</td>
</tr>
<tr>
<td>Vigenère</td>
<td>7.9871</td>
<td>7.9969</td>
<td>7.99871</td>
</tr>
<tr>
<td>Chaotic</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 13 contains the results for entropy or randomness in all the encrypted images. DES, AES and 3-DES have high randomness but chaotic has the highest amongst all.
5.6. PSNR Analysis

Table 14. PSNR Analysis

<table>
<thead>
<tr>
<th></th>
<th>128x128</th>
<th>256x256</th>
<th>384x384</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSNR</td>
<td>8.5915</td>
<td>8.47027</td>
<td>9.2545</td>
</tr>
<tr>
<td>DES</td>
<td>8.45439</td>
<td>8.45200</td>
<td>9.2393</td>
</tr>
<tr>
<td>3DES</td>
<td>8.3620</td>
<td>8.41306</td>
<td>9.2147</td>
</tr>
<tr>
<td>AES</td>
<td>8.29485</td>
<td>8.35793</td>
<td>9.1092</td>
</tr>
<tr>
<td>RC4</td>
<td>7.9074</td>
<td>7.9017</td>
<td>7.2939</td>
</tr>
<tr>
<td>Elliptical</td>
<td>3.1685</td>
<td>3.1359</td>
<td>3.1698</td>
</tr>
<tr>
<td>Visual</td>
<td>3.0984</td>
<td>3.0045</td>
<td>3.1145</td>
</tr>
<tr>
<td>Hierarchical</td>
<td>3.3554</td>
<td>3.3606</td>
<td>3.3649</td>
</tr>
<tr>
<td>Idea</td>
<td>51.1358</td>
<td>51.1171</td>
<td>51.1478</td>
</tr>
<tr>
<td>Vigenère</td>
<td>9.77257</td>
<td>9.78182</td>
<td>10.1096</td>
</tr>
<tr>
<td>Chaotic</td>
<td>8.4211</td>
<td>8.47</td>
<td>9.2462</td>
</tr>
</tbody>
</table>

PSNR analysis is shown by table 14. For a good encryption, low values of PSNR are appreciated. Visual and Hierarchical cryptography have the lowest PSNR values but they deal with the black and white images while all other techniques are working on grayscale images. Data is lost considerably when the image is converted to black and white and hence AES has good PSNR value closely followed by chaotic. Elliptic curve cryptography shows the best PSNR results.

5.7. Computational Time Analysis

Table 15. Encryption Time Analysis

<table>
<thead>
<tr>
<th>Encryption time</th>
<th>128x128</th>
<th>256x256</th>
<th>384x384</th>
</tr>
</thead>
<tbody>
<tr>
<td>DES</td>
<td>11.91755</td>
<td>48.8177</td>
<td>107.7703</td>
</tr>
<tr>
<td>3DES</td>
<td>35.25612</td>
<td>140.5683</td>
<td>344.148</td>
</tr>
<tr>
<td>AES</td>
<td>40.90502</td>
<td>168.47</td>
<td>365.0551</td>
</tr>
<tr>
<td>RC4</td>
<td>0.058099</td>
<td>0.229511</td>
<td>0.576165</td>
</tr>
<tr>
<td>Elliptical</td>
<td>50.87072</td>
<td>200.4698</td>
<td>465.5531</td>
</tr>
<tr>
<td>Visual</td>
<td>0.090859</td>
<td>0.346871</td>
<td>0.791562</td>
</tr>
<tr>
<td>Hierarchical</td>
<td>0.3094</td>
<td>1.111671</td>
<td>2.5075</td>
</tr>
<tr>
<td>Idea</td>
<td>1.1699</td>
<td>4.674032</td>
<td>10.16392</td>
</tr>
<tr>
<td>Vigenère</td>
<td>0.006247</td>
<td>0.018642</td>
<td>0.045694</td>
</tr>
<tr>
<td>Chaotic</td>
<td>0.79132</td>
<td>3.268887</td>
<td>7.333148</td>
</tr>
</tbody>
</table>

Encryption should be done in least time and hence visual and hierarchical cryptographies are better than others as shown in table 15. But Vigenère proves to be the fastest amongst all.

6. Overall Conclusion

This paper is an effort to provide a detailed comparison of various image encryption mechanisms. Ciphers ranging from traditional ciphers to more recent ones such as based
on chaotic encryption mechanism are compared in detail. After implementation of all said schemes, a detailed comparison was drawn. Table 16 provides the whole gist of the results, where all the schemes are given a particular score based on performance. Chaotic method emerges out to be the best amongst all on the basis of overall score.

- Chaotic technique provides high randomness as depicted by the high value of entropy.
- Chaotic technique has a very high sensitivity to a single bit change in original pixel or key.
- PSNR value of hierarchical and visual cryptography is lowest but these techniques work on black and white image and data is lost while conversion, so for grayscale and coloured images AES, RC4 shows better results closely followed by Chaotic.
- Pixels are highly uncorrelated in chaotic technique which again makes it best amongst all.
- Large key space increases the brute force search time and further enhances the security in chaotic mechanism.
- Uniform histograms are also its characteristic feature which make chaotic mechanism stand on top among all.

**Table 16. Overall Comparison**

<table>
<thead>
<tr>
<th>Techniques Implemented</th>
<th>Computational time</th>
<th>Entropy</th>
<th>PSNR</th>
<th>Key sensitivity</th>
<th>Pixel sensitivity</th>
<th>Correlation analysis</th>
<th>Key space</th>
<th>Histogram analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td>DES</td>
<td>Very poor</td>
<td>Good</td>
<td>Good</td>
<td>Best</td>
<td>Very poor</td>
<td>moderate</td>
<td>Very poor</td>
<td>Moderate</td>
</tr>
<tr>
<td>3-DES</td>
<td>Very poor</td>
<td>Good</td>
<td>Good</td>
<td>Best</td>
<td>Very poor</td>
<td>Moderate</td>
<td>Poor</td>
<td>Moderate</td>
</tr>
<tr>
<td>AES</td>
<td>Very poor</td>
<td>Good</td>
<td>Good</td>
<td>Best</td>
<td>Very poor</td>
<td>Moderate</td>
<td>Poor</td>
<td>Good</td>
</tr>
<tr>
<td>RC4</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td>Best</td>
<td>Best</td>
<td>Good</td>
<td>Moderate</td>
<td>Good</td>
</tr>
<tr>
<td>Elliptical</td>
<td>Very poor</td>
<td>Poor</td>
<td>Best</td>
<td>Moderate</td>
<td>Moderate</td>
<td>Good</td>
<td>Best</td>
<td>Poor</td>
</tr>
<tr>
<td>Visual</td>
<td>Good</td>
<td>Poor</td>
<td>Best</td>
<td>Moderate</td>
<td>Moderate</td>
<td>Good</td>
<td>Best</td>
<td>Poor</td>
</tr>
<tr>
<td>Hierarchical</td>
<td>Moderate</td>
<td>Poor</td>
<td>Best</td>
<td>Good</td>
<td>Moderate</td>
<td>Good</td>
<td>Best</td>
<td>Poor</td>
</tr>
<tr>
<td>IDEA</td>
<td>Poor</td>
<td>Very poor</td>
<td>Very poor</td>
<td>Very poor</td>
<td>Poor</td>
<td>Poor</td>
<td>Very poor</td>
<td>Very poor</td>
</tr>
<tr>
<td>Vigenère</td>
<td>Best</td>
<td>Moderate</td>
<td>Good</td>
<td>Poor</td>
<td>Very poor</td>
<td>Very poor</td>
<td>Poor</td>
<td>Very poor</td>
</tr>
<tr>
<td>Chaotic</td>
<td>Moderate</td>
<td>Best</td>
<td>Good</td>
<td>Best</td>
<td>Best</td>
<td>Best</td>
<td>Best</td>
<td>Best</td>
</tr>
</tbody>
</table>

All the techniques discussed in the paper are in spatial domain. Every technique has some advantages and disadvantages but chaotic techniques proved to be more secure and hence in future these may be combined with some frequency domain technique to further obtain better results in terms of enhanced robustness against attacks and reduced time complexity.
References
