Volume 4, Issue 9, September 2014 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Security Algorithm for Data Transfer Using Bmp Image File as Carrier

Abhishek Nayyar
Department of Computer Science, NIT,
Jalandhar, India

Abstract— This publication presents a technique of data transfer by encrypting data in BMP image file without distorting the image. Each pixel value in a BMP image file can be represented in A888RGB 32 bit format and value of each pixel corresponds to some colour of image. This algorithmic approach is based on encryption and decryption of text in BMP image file using some shared secret key. Most of the algorithm in the field of steganography deals either with the original image modification or image compression, but in this paper we will present the design approach of embedding data in image without changing its original texture. The distortion factor (DF) which governs the distortion of image is inversely related to the pixel size of image. Distortion factor can be made negligible by increasing the pixel size, thereby making it a safe and secure algorithm for data transfer.

Keywords— Distortion factor, maskable bit, range qualifier, maskable shift, secret key.

I. INTRODUCTION

Bitmap image file (BMP) is a raster graphic image file format used to store bitmap digital image independent of the display device (like graphic adapter). Bitmap image can be roughly categorized into two sections:

A. Header: First 14 bytes and a fixed segment (DIB header) constitute the header section of BMP image. This contains general information about the BMP file such as its pixel format, type (RGB or BGR) etc.

B. Data: It contains the value of pixel represented either in 24 bit format or 16 bit colour format. Additionally the first 8 bit of BMP image contains the value alpha which controls the display intensity of any image with FF representing complete visibility and 00 denoting no visibility. Each hexadecimal value corresponds to a single pixel value. This hexadecimal value existing in any format can be easily converted in A888RGB format after application of bitwise operations. This conversion is not mandatory for image raster on display window but to maintain uniformity of this algorithm we will consider each pixel in A888RGB (32 bit) format with A representing alpha value and RGB corresponds to different colours. For Example 0xFF00FF00 corresponds to green colour pixel .Combinations of different colour pixel forms a complete BMP image.

.

II. OVERVIEW

We can use each pixel value to store our data without creating any difference in appearance of image. For example value 0xFFFF00 and 0xFFFE00 will almost represent the same colour (nearly red). We can use this difference of bit between these two colours to store our information without change in appearance of our image as two values nearly represents the same colour. Our algorithm will do modifications in pixel value which doesn't affect colour to store our data using a secret key. The final image can be transferred to receiver which will use same secret key to decrypt the data from the pixel values of BMP image. The detail description of Encryption and Decryption of our security algorithm is as below:

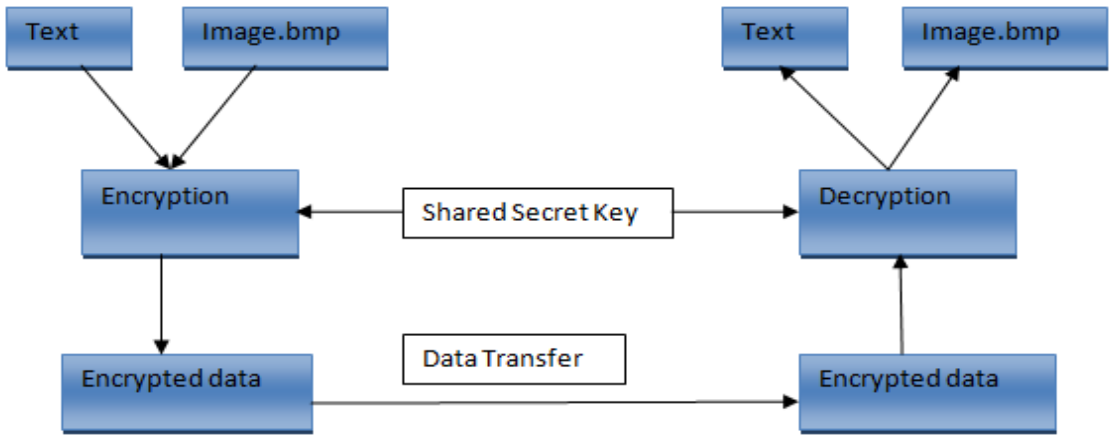


Fig 1. Data transfer stages

III. EXISTING ALGORITHMS TO PROBLEM

There is wide area of algorithms available for image compression and image modification. For example Text and image encryption and decryption using AES (Advanced encryption standard) [1], Rubik cube image encryption which involved scrambling of image by application of some bitwise XOR operations [2]. Image encryption based on block based symmetric transformation algorithm [3]. The following method of encryption / decryption results in the complete modification of image and mainly focus on compression methodology. In the paper we will present a design approach in which original texture of image is not modified even after the text encryption.

IV. DATA ENCRYPTION

Before starting the process of data encryption we assume that secret key which is used during the encryption and decryption mechanism is shared between both the sender and receiver. Firstly we will derive hexadecimal value corresponding to each pixel in A888RGB format in image (I1). Our task is to encrypt text T1 in image I1 using the shared secret key S1.

Let's say we have a text (T1) as "*My name is Abhishek"* .We will look into the method of encrypting this text T into our BMP image (I1) (24 width and 18 height).

```
/* 24 * 18 Pixel */
const unsigned long images[432] = {
0x00DBEBF0,0x00F7F9F8,0x00DDDDDD,0x00EFEFEF,0x00F5F4F4,0x00FDF9F1,0x00F8F5F5,0x00F7F6F3,0x00F8F5F3,0x00F8F4F3,
0x00F4F5F5,0x00F5F4F3,0x00F7F2F0,0x00F4F3F5,0x00FAF3EF,0x00FAF7F4,0x00FCF6F3,0x00FAF6F4,0x00FBF4F4,0x00FBF5F4,
0x00FBF6F2,0x00F6F5F2,0x00F4F4F1,0x00F6F5F3,0x00C9EDFD,0x00EFF3F6,0x00DEDEDD,0x00EEEDEC,0x00F5F6F9,0x008FBEE1,
0x00C1E0EC,0x00B6D4E9,0x00C7E2F2,0x00D4E5F5,0x00DBE4F2,0x00E1EDF7,0x00E2EFF5,0x00F8F9F7,0x00C9E4F0,0x00A0CDEB,
0x00A7CEEB,0x00ADD3EF,0x00B0D8EE,0x00AED6EB,0x00A7D0EA,0x00C9DEEF,0x00E6F2F5,0x00C9E3F5,0x0081B8EB,0x00ECF0F3,
0x00DFDEDE,0x00EEEDEB,0x00EAF1F5,0x007BC2F4,0x008ECDF4,0x0088CCF6,0x009CD1F2,0x00A5D9F8,0x00B5DEF6,0x00DCECF5,
0x00F6FBFC,0x00B1DFFA,0x004FACED,0x0042A1F2,0x0041A6EE,0x003D9FED,0x0066B4F5,0x0096CFF7,0x00C8E7F9,0x00EDFFFD,
0x00E2F4FC,0x00B8DFF8,0x0063AAE4,0x00ECF0F3,0x00E0DEDE,0x00EFEDEB,0x00EAF0F4,0x004EADED,0x006FBCED,0x0075C0F4,
0x0083C4F1,0x00A0CFF1,0x00F6FBFD,0x00FFFFFE,0x00D8EFFC,0x00A2D8F6,0x00C3E7FB,0x00E2F9FD,0x00F2FFFE,0x00E3F9FF,
0x00D6EEFC,0x00E2EFF4,0x00FBFEFA,0x00DBF3FE,0x009ECCEC,0x00AFDDF4,0x009BDEF9,0x00EBEEF3,0x00DEDDDC,0x00EDEBE9,
0x00E9F0F4,0x0063BBF6,0x0074C2F5,0x008CCBF5,0x00C7E9F4,0x00EFF8FC,0x00FFFFFE,0x00FFFFFA,0x00F6FFFC,0x00FFFFFD,
0x00EBF7FE,0x00C7E4F5,0x00C9EAFA,0x00D0F0FD,0x00D0EBFA,0x00CDE8F8,0x0098CEF1,0x0084C2EE,0x006FB1E4,0x006FB0E5,
0x008AD6FE,0x00E7EDF3,0x00DDDCDB,0x00ECEBEA,0x00EEF0F0,0x00ABDEF9,0x00DA2D8F7,0x00A3DAF9,0x00ACDCFD,0x00A6DBF8,
0x00C1E2F7,0x00E4F3FD,0x00E7F2FE,0x00D6ECFA,0x00C8E6F5,0x00CADCE4,0x00EBF9FF,0x00D5E5EB,0x00C2E5F4,0x00BCE5F9,
0x00AFDEF9,0x008DD1F7,0x0087CFFC,0x009DD8F9,0x005E8000,0x00E8E9DD,0x00DDDDDF,0x00EBEAEA,0x00EDECEF,0x00C9E6FF,
0x00D0EAFF,0x00CBEAFF,0x00CCEBFE,0x00D9F6FF,0x00D7F4FF,0x00D7F3FF,0x00C5DCF1,0x00CAD8D4,0x00F4FEFF,0x00EAF9FF,
0x00B8BDC1,0x00C9E0F1,0x00BDE5FF,0x00CCF1FF,0x00CCEFFF,0x00C2EBFF,0x009ADDFF,0x007BD3FF,0x004D7D00,0x00E6E8E1,
0x00DDDCDE,0x00EAEAEC,0x00E8EAE0,0x0069A800,0x0075AC29,0x007CB12E,0x006CA40F,0x00679A06,0x006DA30B,0x0071AB0F,
0x0087AA32,0x007E5F2D,0x008CC257,0x00A5CF88,0x008F7948,0x00777351,0x00A5D0B6,0x008CC691,0x0092C98D,0x0090C182,
0x0085AF5E,0x00729728,0x003B6E00,0x00E4E5E2,0x00DCDCDD,0x00EAE9ED,0x00E7E9E1,0x0070B600,0x007CBD0D,0x0076B90E,
0x0079BB11,0x007BBB12,0x007ABA12,0x007ABC13,0x0056B010,0x004C5611,0x0071B114,0x007BB31C,0x00916E2D,0x00685F1E,
0x0067A007,0x00649E00,0x0065A100,0x00679C01,0x00609305,0x005D8F07,0x003A6C00,0x00E2E3E1,0x00DBDBDC,0x00E9E8EB,
0x00E5E6DF,0x0066A900,0x0070AC0A,0x006CA60E,0x006BA60D,0x0069A40B,0x0069A409,0x006BA707,0x006EB007,0x006EB509,
0x0068A00C,0x003F6E15,0x00403912,0x00333409,0x004D860A,0x00548C08,0x00558609,0x00528406,0x004F8206,0x00508106,
0x002D5B00,0x00E2E3DF,0x00DBDADC,0x00E9E8EC,0x00E3E5DD,0x00579800,0x00629D04,0x00629E05,0x00629D06,0x00649D0B,
0x00649D0A,0x005F9807,0x005C9505,0x005A9203,0x00558E02,0x00589304,0x00549108,0x00559008,0x004B8208,0x004A8106,
0x004C7E05,0x004A7D05,0x00477A04,0x00477804,0x006B8948,0x00D5DAD4,0x00DBDADC,0x00E8E7EB,0x00E0E2DB,0x004A8400,
0x00568D03,0x00528B05,0x00548C05,0x00578C07,0x00548904,0x00528604,0x004F8409,0x00518405,0x00538510,0x00457806,
0x00417400,0x003B7106,0x00D6DF04,0x0034A6D0,0x00457009,0x003E6D01,0x00446E08,0x0048730F,0x00688A4D,0x00D6DAD5,
0x00DAD9DB,0x00E9E8EB,0x00DEE0D9,0x00376F00,0x00497D02,0x004C7DD6,0x004E8007,0x004D8005,0x004A7E05,0x00437803,
0x00407202,0x00356400,0x00A0BA86,0x0094AD73,0x003B6A0E,0x0062883F,0x00829A66,0x00214800,0x00ABC291,0x00325706,
0x00819962,0x0080945E,0x001F4B00,0x00DCDEDC,0x00D9D9DA,0x00E7E7EA,0x00DCDED8,0x00295900,0x003F6C04,0x00396804,
0x003A6903,0x003B6903,0x003E6E03,0x003E7002,0x003E6D02,0x00305F00,0x00988B37C,0x0091B177,0x00335F09,0x00779456,
0x0097AD7B,0x0048681D,0x00A0B887,0x004A6C24,0x00768F59,0x00829B67,0x001D4200,0x00D7DCD1,0x00D9D8DA,0x00E8E7E9,
0x00DBDCD5,0x00214D00,0x00355E01,0x00356306,0x00336105,0x00305D06,0x00335F05,0x00335D03,0x00345E03,0x00325D00,
0x008EA970,0x00516F1D,0x00779650,0x008FA96F,0x005D7F2F,0x00A6BA8C,0x00567624,0x006C8741,0x00839C5B,0x00406213,
0x00D8DCD5,0x00E3E3E9,0x00D5D4D4,0x00E7E6E7,0x00D6D9D1,0x000E3A00,0x00234C00,0x00285500,0x00245100,0x00295800,
0x00295600,0x00295600,0x00285100,0x007E9F55,0x0081A057,0x00375E00,0x00719641,0x00265400,0x00718F47,
0x00295500,0x005E8132,0x005C7C30,0x00305500,0x00DAD7DA,0x00D6D5D3,0x00E4E4E4,0x00E3E3E4,0x00E3E3E3,0x00D7DDD4,
0x00DADFD7,0x00D7DCD6,0x00D9DED8,0x00D7DCD6,0x00D9DFD8,0x00DADFD7,0x00D9DED6,0x00D9DED4,0x00D0D6CD,0x00D0D7CE,
0x00D5DCD3,0x00D0D7CE,0x00D5DBD5,0x00D0D6CD,0x00D7DAD4,0x00D4D7D1,0x00D7D7D3,0x00DADBD6,0x00000000,0x00000000,
0x00000000,0x00EBEBEB,0x00D5D5D5,0x00D9D8D8,0x00D9D8D8,0x00D9D8DB,0x00D9D8D9,0x00D9D8D9,0x00D9D8D8,0x00D9D8D8,
0x00D9D8DB,0x00D9D8DD,0x00D9D8DB,0x00D9D8DA,0x00D9D8D8,0x00D9D8D8,0x00DAD8D9,0x00D9D8D9,0x00DAD7DB,0x00DAD8DA,
0x00D9D8DA,0x00D9D8D8
};
```

Fig 2. Hexademicimal A888RGB format representation of image I1

Fig 3. Image file before encryption

Step 1: Sub Key Generation

For example for the given bmp image value of our secret key would be S1 = 0x00424DB8 which is shared between both the sender and receiver. Now s1 can be broken down further into three different keys :

S11 = S1 & 0x00FF0000 (masker, which is a fixed value used for three different key generation)

S12 = S1 << 8 & masker

S13 = S1 << 16 & masker

S11 = 0x00420000, S12 = 0x004D0000, S13 = 0x00B80000

Now we have value of our secret key S1 and three sub keys S11, S12, S13 which will be used further in encryption algorithm where S11 represents location of a pixel value which would be used as maskable bit, S12 represents the location of pixel at which length of text to be encoded will be stored and S13 represents range qualifier (Description of all in later section).

Step 2: Find out the location of maskable bit and storage location of length of text

Maskable bit M1 is the hexadecimal value which will be used as modifier key in encryption of text into image. Value of the maskable bit (M1) would be the pixel value (hex value) at decimal equivalent of value S11 in the pixel range. For example, decimal equivalent of S11 in pixel range is dec (S11) % (image width * image height).

Dec (S11) = 4325376 % (24 * 18) = 192.

We have got a location value 192 in 432 length pixel array of image at which our maskable bit is stored. Now it is the possibility that when we calculate the position to store the data it may coincide with position of maskable bit, as the result values may get modified. This will lead to ambiguity during the time of decryption of image. To avoid such conflicts between the indicator values and actual data values, we can divide our hexadecimal array of image into two sections. On section above a fixed location will store the indicator value and other section below the fixed location will store text. The fixed value described here is **range qualifier,** which is the decimal value of security key S13. For our BMP image the value of range qualifier (S13) would be dec (S13) % (image width * image height).

RQ = 12058624 % (24 * 18) = 208.

Value of location of M1 would be the dec (s11) in range above RQ i.e. in this bmp image values above position 208 would be indicator values such as maskable bit and location of length of text and values below 208 will depict actual text value. To calculate the value of Loc (M1) we have to find value of dec (S11) in RQ range. For that we would check the rightmost first non zero bit of two numbers dec (S11) and RQ. If position of first rightmost non zero bit for both the numbers is same then final location Loc (M1) would be

Loc (M1) = dec (s11) ^ RQ

If the position of rightmost non zero bit of two numbers is not same then

Loc (M1) = (value1 >> (difference between the positions of rightmost non zero bit)) ^ value2.

Where value 1 is greater value between two values. In our BMP image location of maskable bit M1 will be

Loc (M1) = 16 and value of maskable bit would value at position 16 in array. So our maskable bit M1 is 0x00FCF6F3.

Now our next step is to find the length of text and store it at some pixel value. This will be used at the time of encoding or decoding of image. The position at which the length of text will be stored is determined by the sub secret key S12. It is calculated by finding the decimal value of S12 in given range.

Dec (S12) = S12 % (image width * image height) => 5046272 % (24 * 18) => 80

Loc (Len (T1)) = Dec (S12) ^ RQ => 80 ^ RQ => 80

As we got the value of position at which length of text would be stored, copy the value at Loc (Len (T1)) to its next position Loc (Len (T1)) + 1, which will be used at the time of decrypting image. Find the hexadecimal equivalent of length of text (hex ((Len (T1)) and XOR it with pixel value at Loc (Len (T1))

Pixel (Loc (Len (T1)) + 1) = Pixel (Loc (Len (T1)))

Pixel (Loc (Len (T1))) = Pixel (Loc (Len (T1))) ^ (hex ((Len (T1))

Now we are ready to encrypt the text in image with maskable bit M1 and length of text encrypted in image.

Step 3: Encrypting text in BMP image

Text encryption in BMP image is based on the value of maskable bit (M1). M1 is modified each time to get a unique position value in the given range of data for each character of text. In calculating the position value for each character of text, it's important to get the unique positions which **are not adjacent to each other.** Unique position for each character will store pixel value formed after encrypting existing pixel value with the text value and its adjacent pixel value will be value of previous pixel before modification which will act as recovery value of character. So if two positions for character storage are adjacent then its high probability that recovery value of one character get modified by the actual value of another character. This will create problems at time of decryption and we will not be able to retrieve the original text.

The key concept behind the encryption of text is to calculate a unique position corresponding to each character in text, then copy the existing pixel value at that location to the adjacent location and then modify it to store the value corresponding to the ASCII value of character. The complete algorithm is key step of calculating a unique position corresponding to each character in text irrespective of value of character. In calculation of unique position an important criteria which must be fulfilled is to keep a check that adjacent position to the unique position must not be the unique position for another character in text. If this condition fails than recovery value for each character gets overwritten by some another value, which may lead to anomalies at the time of decoding of text. Complete process of text encryption in image is based on two sections.

3.1 Calculation of unique position corresponding to each character of text

Series of modification in maskable value leads to a position which is unique for each character. Before making any calculations we are sure about availability of two important data value calculated in previous steps available with us, one is value of maskable bit and another is length of text. To perform this algorithm we perform following steps.

1. Take maskable bit, copy it in some temporary variable and all the action corresponding to single character are performed on that maskable bit. OR maskable bit with position value of character in text. For example for first letter L we will OR maskable bit with one. Similarly with two, three for second and third character and so on. [Note: For performing any operating on maskable bit, we must make **sure that value of original maskable bit doesn't get modified** because after performing series of modifications on maskable bit we will get a number corresponding to character and for next character we will start with value of original maskable bit. So we must perform modification **operation on copy of maskable bit rather** than using original value]
2. Shift the resultant maskable bit by positions equal to value of variable masking shift (ms). Masking shift is the variable which has the value equal to the position value of character in text (T) if position is less than 32; otherwise it's equal to position of character mod (%) 32. As we are performing our calculations of 32 bit hex number so

maximum possible shifts in any number can be 31. So the minimum and maximum value of ms is 1 and 31. At the end of this step we get modified maskable bit M (M1) after OR and shift.

3. Binary value B(M1) corresponding to modified maskable bit M(M1) is calculated ,and then maximum limit range check operation is performed on the resulting value i.e. B(M1) mod(%) (Image width and image height). The resulting value is added with the value corresponding to the position of character in text (T1).
4. Resulting value from the previous step is processed through the condition of range qualifier (RQ) which is binary value corresponding to sub secret key S13. RQ condition in this processing differs from the RQ condition processing in main step 1 and 2 i.e. steps for maskable bit and position for storage of length of text calculation. The difference is that the value of previous section corresponds to support value so it must be below value of RQ position whereas value corresponding to this section is data value so it must be above RQ position. This is done by performing OR operation on RQ and resulting B (M1) value. This will ensure that our data storage position is above the position of RQ and below maximum limit which is ensured by max range check.
5. The resulting value may or may not be unique. We have to insure that position value for each character is unique and even its adjacent position is also not occupied by position value for another character. For this we perform following steps
   a. Let say we get final value F (M1)1 for the first character of text after performing steps till 4. We will check the alpha value of pixel which is last eight bytes in A888RGB format pixel BMP. If the value at this position and adjacent position both are zero (by default its zero, if not we can set it to zero at start of our encryption).Then we need not to modify F (M1)1 which will be the final position at which the text will be stored. We will simple change the alpha value at position and adjacent to 0F.
   b. And if alpha value at any of the current position or next position is not zero. It implies that it's being used by another character, so we will keep on incrementing the value F (M1)1 till the condition for current and next position mentioned in step (a) is satisfied. Now at this current position and adjacent position we will change the value of alpha to 0F

For performing encryption of each character in text we will start each time with original maskable bit and modify it to get a unique position value in the end, which defines the location at which the character would be stored.

3.2 Storing the value of text at unique position

Once we get the unique position (UP1) corresponding to each character in text our next task is to store the value of character of text at that position. For this we will first transform our character to hexadecimal form by performing OR operation with 0x00000000. Let's say first character in our text T be Tx1

Tx1 = Tx1 | 0x00000000

Before performing any modification at the position (UP1) we will store its value at (UP1) + 1 i.e. its adjacent position. Now XOR the pixel value at (UP1) with value Tx1.

Pixel (UP1) = Pixel (UP1) ^ Tx1

Now we have got the final position corresponding to single character in text at which value of character is stored. We will perform the same steps corresponding to each character in text and in end we will get a modified image with encrypted data.

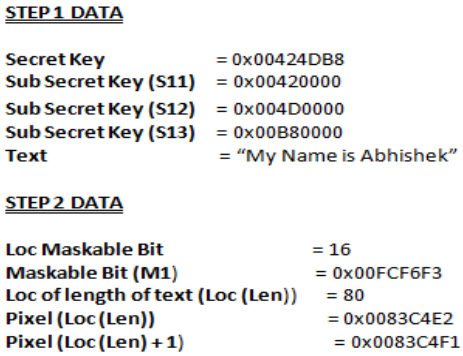


Fig 4. Stat summary

Table I: Modified Values Using Values In Fig 3

S.No	Character	Position	Original Px	Original + 1	Modified Px	Modified + 1
1	M	215	0x005d8f07	0x003a6c00	0x005d8f4a	0x005d8f07
2	y	222	0x0070ac0a	0x006ca60e	0x0070ac73	0x0070ac0a
3		251	0x005f9807	0x005c9505	0x005f9827	0x005f9807
4	N	244	0x00e3e5dd	0x00579800	0x00e3e593	0x00e3e5dd
5	a	213	0x00679c01	0x00609305	0x00679c60	0x00679c01
6	m	217	0x00e2e3e1	0x00dbdbdc	0x00e2e38c	0x00e2e3e1
7	e	247	0x00629e05	0x00629d06	0x00629e60	0x00629e05
8		219	0x00e9e8eb	0x00e5e6df	0x00e9e8cb	0x00e9e8eb

9	I	224	0x006ba60d	0x0069a40b	0x006ba644	0x006ba60d
10	s	226	0x0069a409	0x006ba707	0x0069a47a	0x0069a409
11		228	0x006eb007	0x006eb509	0x006eb027	0x006eb007
12	A	230	0x0068a00c	0x003f6e15	0x0068a04d	0x0068a00c
13	b	253	0x005a9203	0x00558e02	0x00a9261	0x005a9203
14	h	255	0x589304	0x00549108	0x0058936c	0x00589304
15	i	232	0x00403912	0x00333409	0x0040397b	0x00403912
16	s	208	0x00916e2d	0x00685f1e	0x00916e5e	0x00916e2d
17	h	241	0x00e2e3df	0x00dbdadc	0x00e2e3b7	0x00e2e3df
18	e	210	0x0067a007	0x00649e00	0x0067a062	0x0067a007
19	k	249	0x00649d0b	0x00649d0a	0x00649d60	0x00649d0b

As changed pixel value very closely resembles the value of previous pixel, so no much change in image value is depicted. Also during the time of decryption intruder will not have the original image to perform any comparison function as we will only be sending the encrypted image to the receiver. So chance for identification of any modifications in image is negligible. As there is no change in alpha value in final image prepared for sending so to make out for actual encrypted area become more difficult as the only way to decrypt is through the use of secret key. So the security and mechanism for secret key generation becomes very critical to this algorithm. It's mechanism for secret key transfer may or not be inbuilt in the image as far as level of security is concerned but as far as this mechanism is concerned it deal with the transfer of secret key and secret key generation in built in the image. This makes overhead of any redundant data very minimal in this approach.

```
unsigned long . . sample[432] = {
0x00DBEBF0,0x00F7F9F8,0x00DDDDDD,0x00EFEFEF,0x00F5F4F4,0x00FDF9F1,0x00F8F5F5,0x00F7F6F3,0x00F8F5F3,0x00F8F4F3,0x00F4F5F5,0x00F5F4F3,0x00F7F2F0,0x00F4F3F5,
0x00FAF7F4,0x00FCF6F3,0x00FAF6F4,0x00FBF4F4,0x00FBF5F4,0x00FBF6F2,0x00F6F5F2,0x00F4F4F1,0x00F6F5F3,0x00C9EDFD,0x00EFF3F6,0x00DEDEDD,0x00EEEDEC,0x00F5F6F9,
0x008FBEE1,0x00C1E0EC,0x00B6D4E9,0x00C7E2F2,0x00D4E5F5,0x00D6E4F2,0x00E1EDF7,0x00E2EFF5,0x00F8F9F7,0x00C9E4F0,0x00A0CDEB,0x00A7CEEB,0x00ADD3EF,0x00B0D8EE,
0x00AED6EB,0x00A7D0EA,0x00C9DEEF,0x00E6F2F5,0x00C9E3F5,0x0081B8EB,0x00ECF0F3,0x00EEEDEB,0x00EAF1F5,0x007BC2F4,0x008ECDF4,0x0088CCF6,0x009CD1F2,0x00A5D9F8,
0x00B5DEF6,0x00DCECF5,0x00F6FBFC,0x00B1DFFA,0x004FACED,0x0042A1F2,0x0041A6EE,0x003D9FED,0x0066B4F5,0x0096CFF7,0x00C8E7F9,0x00EDFFFD,0x00E2F4FC,0x00B8DFF8,
0x0063AAE4,0x00ECF0F3,0x00E0DEDE,0x00EFEDEB,0x00EAF0F4,0x004EADED,0x006FBCED,0x0083C4E2,0x0083C4F1,0x00F6FBFD,0x00FFFFFE,0x00D8EFFC,0x00A2D8F6,0x00C3E7FB,
0x00E2F9FD,0x00F2FFFE,0x00E3F9FF,0x00D6EEFC,0x00E2EFF4,0x00FBFEFA,0x00DBF3FE,0x009ECCEC,0x00AFDDF4,0x009BDEF9,0x00EBEEF3,0x00DEDDDC,0x00EDEBE9,0x00E9F0F4,
0x0063BBF6,0x0074C2F5,0x008CCBF5,0x00C7E9F4,0x00EFF8FC,0x00FFFFFE,0x00FFFFFA,0x00F6FFFC,0x00FFFFFD,0x00EBF7FE,0x00C7E4F5,0x00C9EAFA,0x00D0F0FD,0x00D0EBFA,
0x00CDE8F8,0x0098CEF1,0x0084C2EE,0x006FB1E4,0x006FB0E5,0x008AD6FE,0x00E7EDF3,0x00DDDCDB,0x00ECEBEA,0x00EEF0F0,0x00ABDEF9,0x00A2D8F7,0x00A3DAF9,0x00ACDCFD,
0x00A6DBF8,0x00C1E2F7,0x00E4F3FD,0x00E7F2FE,0x00D6ECFA,0x00C8E6F5,0x00CADCE4,0x00EBF9FF,0x00D5E5EB,0x00C2E5F4,0x00BCE5F9,0x00AFDEF9,0x008DD1F7,0x0087CFFC,
0x009DD8F9,0x005E8000,0x00E8E9DD,0x00DDDDDF,0x00EBEAEA,0x00EDECEF,0x00C9E6FF,0x00D0EAFF,0x00CBEAFF,0x00CCEBFE,0x00D9F6FF,0x00D7F4FF,0x00D7F3FF,0x00C5DCF1,
0x00CAD8D4,0x00F4FEFF,0x00EAF9FF,0x00B8BDC1,0x00C9E0F1,0x00BDE5FF,0x00CFF1FF,0x00CCEFFF,0x00C2EBFF,0x009ADDFF,0x007BD3FF,0x004D7D00,0x00E6E8E1,0x00DDDCDE,
0x00EAEAEC,0x00E8EAE0,0x0069A800,0x0075AC29,0x007CB12E,0x006CA40F,0x00679A06,0x006DA30B,0x0071AB0F,0x0087AA32,0x007E5F2D,0x008CC257,0x00A5CF88,0x008F7948,
0x00777351,0x00A5D0B6,0x008CC691,0x0092C98D,0x0090C182,0x0085AF5E,0x0072972B,0x003B6E00,0x00E4E5E2,0x00DCDCDD,0x00EAE9ED,0x00E7E9E1,0x0070B600,0x007CBD0D,
0x0076B90E,0x0079BB11,0x007BBB12,0x007ABA12,0x007ABC13,0x00568010,0x004C5611,0x0071B114,0x007BB31C,0x00916E5A,0x00916E2D,0x0067A006,0x0067A007,0x0065A100,
0x00679C60,0x00679C05,0x005D8F4A,0x005D8F07,0x00E2E38C,0x00E2E3E1,0x00E9E8CB,0x00E9E8EB,0x0066A900,0x0070AC78,0x0070AC0A,0x006BA644,0x006BA60D,0x0069A47A,
0x0069A409,0x006EB027,0x006EB007,0x0068A04D,0x0068A00C,0x0040397B,0x00403912,0x004D860A,0x00548C08,0x00558609,0x00528406,0x004F8206,0x00508106, 0x001D4200,
0x002D5B00,0x00E2E3B7,0x00E2E3DF,0x00E9E8EC,0x00E3E593,0x00E3E5DD,0x00629D04,0x00629E60,0x00629E05,0x00649D60,0x00649D0B,0x005F9827,0x005F9807,0x005A9261,
0x005A9203,0x0058936C,0x00589304,0x00559008,0x004B8208,0x004A8106,0x004C7E05,0x004A7D05,0x00477A04,0x00477804,0x006B8948,0x00D5DAD4,0x00DBDADC,0x00E8E7EB,
0x00E0E2DB,0x004A8400,0x00568D03,0x00528B05,0x00548C05,0x00578C07,0x00548904,0x00528604,0x004F8409,0x00518405,0x00538510,0x00457806,0x00417400,0x003B7106,
0x003D6F04,0x003A6D00,0x00457009,0x003E6D01,0x00446E08,0x0048730F,0x00688A40,0x00D6DAD5,0x00DAD9DB,0x00E9E8EB,0x00DEE0D9,0x00376F00,0x00497D02,0x004C7D06,
0x004E8007,0x004D8005,0x004A7E05,0x00437803,0x00407202,0x00356400,0x00A0BA86,0x0094AD73,0x003B6A0E,0x0062883F,0x00829A66,0x00214800,0x00ABC291,0x00325706,
0x00819962,0x0080945E,0x001F4B00,0x00DCDEDC,0x00D9D9DA,0x00E7E7EA,0x00DCDED8,0x00295900,0x003F6C04,0x00396804,0x003A6903,0x003B6903,0x003E6E03,0x003E7002,
0x003E6D02,0x00305F00,0x0098B37C,0x0091B177,0x00335F09,0x00779456,0x0097AD7B,0x0048681D,0x00A0B887,0x004A6C24,0x00768F59,0x00829B67,0x001D4200,0x00D7DCD1,
0x00D9D8DA,0x00E8E7E9,0x00DBDCD5,0x00214D00,0x00355E01,0x00356306,0x00336105,0x00305D06,0x00335F05,0x00335D03,0x00345E03,0x00325D00,0x008EA970,0x00516F1D,
0x00779650,0x008FA96F,0x005D7F2F,0x00A6BA8C,0x0056762A,0x006C8741,0x00839C5B,0x00406213,0x00D8DCD5,0x00E3E3E9,0x00D5D4D4,0x00E7E6E7,0x00D6D9D1,0x00E3A00,
0x00234C00,0x00285500,0x00245100,0x00295800,0x00295600,0x00295600,0x002D5600,0x00285100,0x007E9F55,0x0081A057,0x00375E00,0x00719641,0x00265400,0x00718F47,
0x00295500,0x005E8132,0x005C7C30,0x00305500,0x00DAD7DA,0x00D6D5D3,0x00E4E4E4,0x00E3E3E4,0x00E3E3E3,0x00D7DDD4,0x00DADFD7,0x00D7DCD6,0x00D9DED8,0x00D7DCD6,
0x00D9DFD8,0x00DADFD7,0x00D9DED6,0x00D9DED4,0x00D0D6CD,0x00D0D7CE,0x00214800,0x00D5DCD3,0x00D0D7CE,0x00D5DBD5,0x00D0D6CD,0x00D7DAD4,0x00D4D7D1,0x00D7D7D3,
0x00DADBD6,0x00000000,0x00000000,0x00000000,0x00EBEBEB,0x00D5D5D5,0x00D9D8D8,0x00D9D8D8,0x00D9D8DB,0x00D9D8D9,0x00D9D8D8,0x00D9D8D8,0x00D9D8D8,0x00D9D8DB,
0x00D9D8DD,0x00D9D8DB,0x00D9D8DA,0x00D9D8D8,0x00D9D8D8,0x00DAD8D9,0x00D9D8D9,0x00DAD7DB,0x00DAD8DA,0x00D9D8DA,0x00D9D8D8
};
```

Fig 5. Modified hexadecimal value of image after encryption

Fig 6. Modified Image after encryption

V. DATA DECRYPTION

After encryption of data in image we modify the alpha value of all pixels back to 00. Now at the receivers end we got the encrypted image, but we can't make out, whether the image is encrypted or not by looking at the pixel value of image as we don't have any original image to compare with because sender has send us only the modified image . Also the alpha value of the entire pixels is same. The only way to decode the image is through the use of secret key. In order to decrypt the image we need to perform following steps.

Step 1: Sub Key generation

This step is similar to that in encryption method in which we will generate the values of three sub secret key from the secret key .At the end we have three values S11, S12, S13

Step 2: Identification of maskable bit and length of text

We will identify the position of maskable bit from S11 in similar way we did at the time of data encryption. Pixel value at that position is our maskable bit. Our next step is identifying the position where the length of text is stored in the similar manner as done during the time of encryption. Now we need to XOR the pixel value at that position and its adjacent position to get the length of text.

Len(T1) = Px(pos(Len(T1))) ^ Px(pos(Len(T1)) + 1)

Step 3: Decrypting the data from image

After performing step 1 and 2 we have got the value of both the maskable bit as well as the length of text, so our next task will be to use that value to decrypt the data encoded in that image. Our process of data decryption is based further of two steps, one is identification of unique position for each character of text which is similar to the step done during the time of data encryption. After we got the value of unique position for each character of text, which will be similar to the values in encryption as same approach and key is used, we need to decrypt it to get back original character.

3.1 Calculation of unique position corresponding to each character of text

This step is completely similar to step done during the time of data encryption. The resulting values of this step will also be similar to the resulting values of step done during the time of data encryption as the mechanism and secret key required in this approach are same. So after we got the correct value of position corresponding to each character in image, our next task will be to get back our original character using the pixel values at that position.

3.2 Decrypting text from image

After we got the value of unique position corresponding to each character in image our next task will be to extract our character value from the pixel at that position. To do this we just need to XOR the value at that position with the adjacent value, which will result in value corresponding to ASCII value of character stored at that position.

ASCII (T11) = Px (Pos (T11)) ^ Px (Pos (T11) + 1)

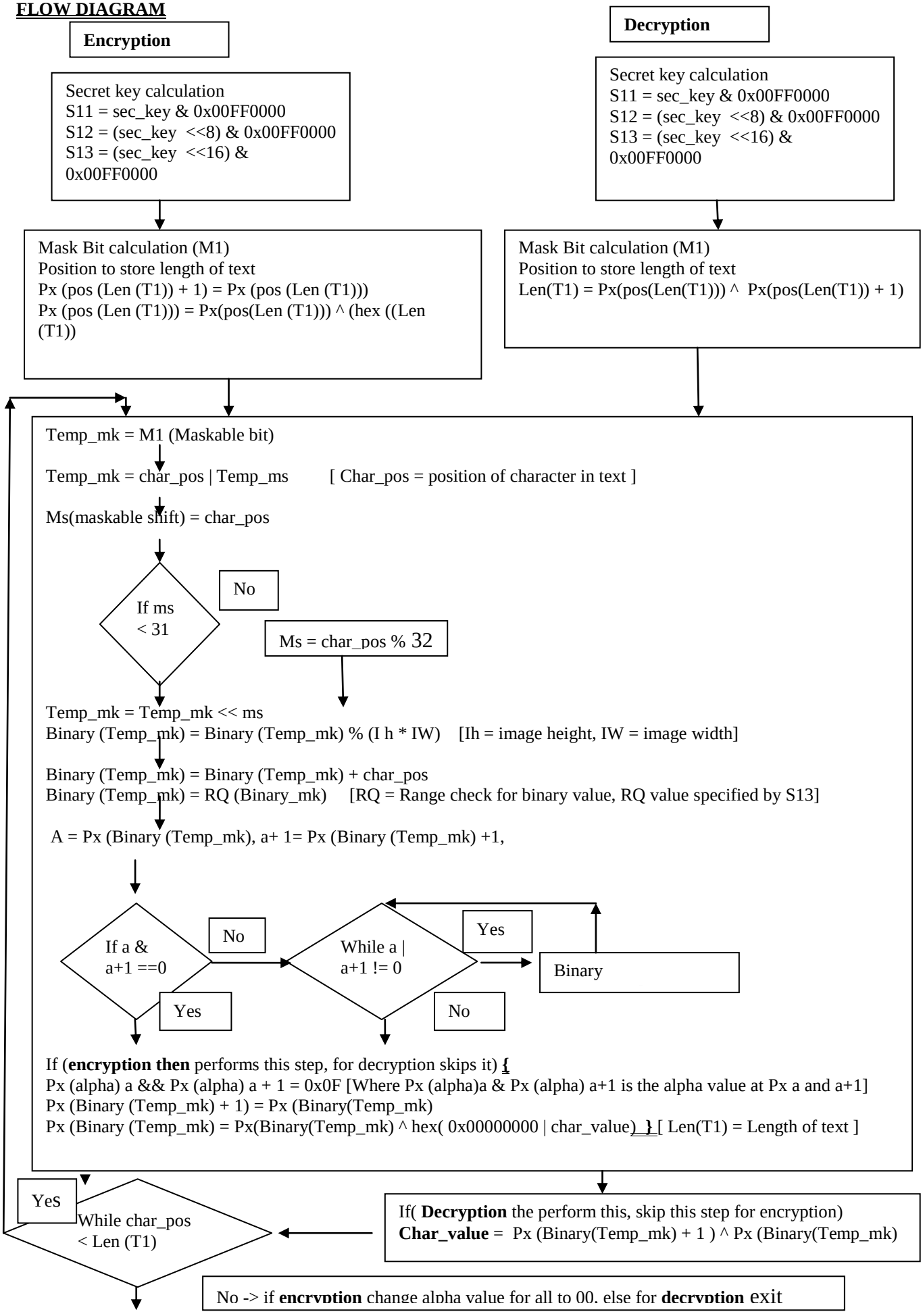
Where Px (Pos (T11)) symbolize the pixel vale at unique position corresponding to character 1 in the text (T1). We need to follow these steps the number of times equal to length of text which we got after step 2. This will result in ASCII value corresponding to each character in text, which can be clubbed together and denoted in string form to get back our original text. This is how we can decrypt our image.

Table II: Text Decryption Summary

S.No	Position	Px (Pos (T11)) ^ Px (Pos (T11) + 1)	Decoded character
1	215	0x005d8f4a ^ 0x005d8f07	M
2	222	0x0070ac73 ^ 0x0070ac0a	y
3	251	0x005f9827 ^ 0x0070ac0a	
4	244	0x00e3e593 ^ 0x00e3e5dd	N
5	213	0x00679c60 ^ 0x00679c01	a
6	217	0x00e2e38c ^ 0x00e2e3e1	m
7	247	0x00629e60 ^ 0x00629e05	e
8	219	0x00e9e8cb ^ 0x00e9e8eb	
9	224	0x006ba644 ^ 0x006ba60d	I
10	226	0x0069a47a ^ 0x0069a409	s
11	228	0x006eb027 ^ 0x006eb007	
12	230	0x0068a04d ^ 0x0068a00c	A
13	253	0x00a9261 ^ 0x005a9203	b
14	255	0x0058936c ^ 0x00589304	h
15	232	0x0040397b ^ 0x00403912	i
16	208	0x00916e5e ^ 0x00916e2d	s
17	241	0x00e2e3b7 ^ 0x00e2e3df	h
18	210	0x0067a062 ^ 0x0067a007	e
19	249	0x00649d60 ^ 0x00649d0b	k

Data decryption restores the original value of text without distorting image. Following table shows the recovery mechanism of text from BMP image file. Value at the current position and adjacent position to the text are XOR in order to recover the original value of text.

FLOW DIAGRAM



VI. EXPERIMENT AND RESULTS

The main aim of this algorithm is to make sure that the image file used for data transfer retains its original texture. This is governed by self induced factor known as **Distortion factor (DF).** Distortion factor is termed as ratio of number of pixel modified to total number of pixel keeping the check on extent of modification factor (K).

Number of pixel modified = 2 * Length of text

Distortion factor (DF) = Number of pixels modified/ (Total number of pixel * K)

Where K is average of difference in original and modified pixel value.

The main aim of this algorithm is to keep DF factor minimum. This can be made negligible by embedding small text in large pixel size image at the average modification rate. Following graph shows DF variation with length and pixel at constant K value.

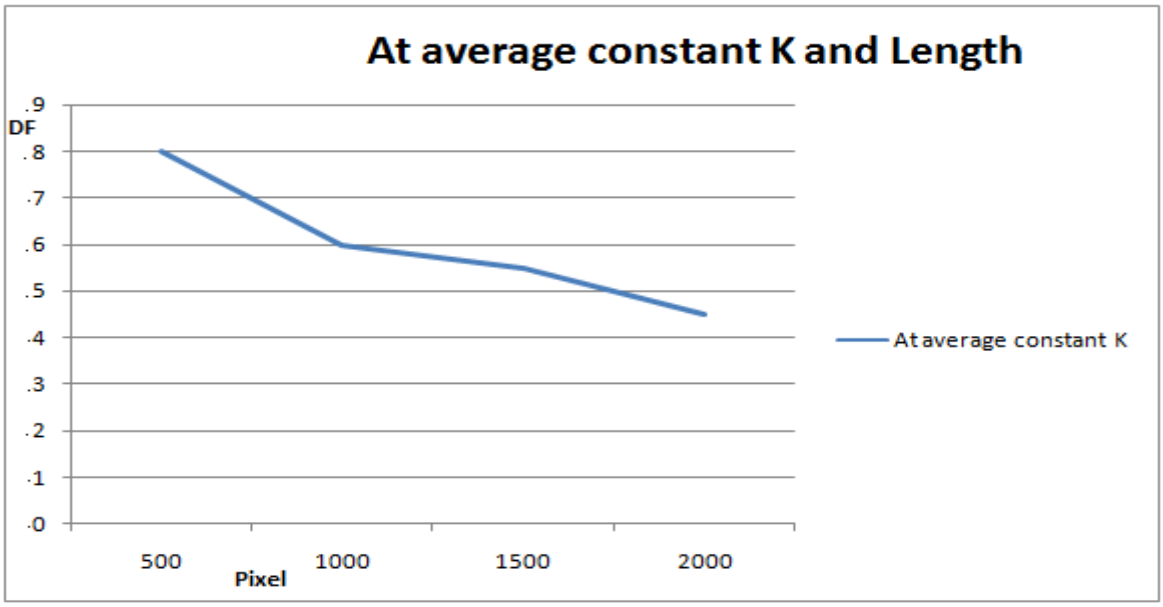


Fig 7. Variation of DF with total number of pixel at constant k and length

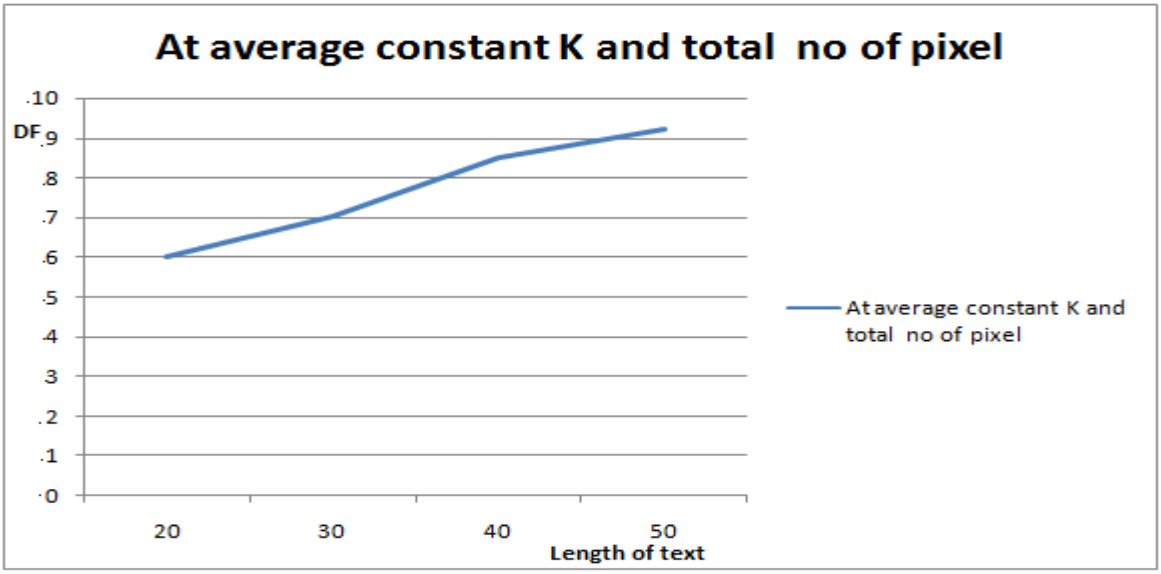


Fig 8. Variation of DF with length of text at constant k and total number of pixels

VII. CONCLUSION

This algorithm for data transfer has added security advantage like; as the modified image is send, so receiver doesn't have the original image to compare with, as the result modification in the pixel value cannot be detected. Value of pixel even after modification resemble same colour so we don't observe much change in the output of image even after modifications. Value of alpha is changed back to original, so it's not possible to decode the image even through the transparency check of pixels in the image. Value of secret key lies in the image itself so the overhead of sending extra data is not there, even though to achieve more security, secret key can be transferred through some shared public key security method without affecting the algorithmic approach. Value of modified pixel is so random and at different location it's not possible to decode it either without the original image or security key, so the possibility of brute force attack in this algorithm is negligible. Following features makes this algorithm a safe algorithm for data transfer.

VIII. FUTURE SCOPE

This algorithm is depicted on a BMP image file, but can be modified for any other file format also like JPG, PNG etc. This can be further extended to encryption of data in video which is also covered by frame rate (picture) with more than 30 frames per second. Data can be encoded in video frames which are nothing but pictures and can be easily transferred. As video has commodious number of frames, which makes it apt for supporting huge data transfer. This can have a promising future in field of security.

ACKNOWLEDGMENT

It gives me a great pleasure to express my deep sense of gratitude and indebtedness to my guide **Mr. Deepak Kumar Gupta**, Associate Professor NIT Jalandhar for his valuable support and encouraging mentality throughout the research work. I also want to express my gratitude to **Dr Arun Khosla,** Dean NIT Jalandhar**, Mr Nishant Singh**, HBIT Kanpur and Mr**. Amit Kumar Verma,** BITS Pilani for their support.

REFERENCES

[1] Kundan Kumar, Amit Kumar Mishra, Text and image encryption and decryption using advanced encryption standard.

[2] Khaled Loukhaoukha, Jean-Yves Chouinard, and Abdellah Berdai, Secure image encryption and decryption using Rubik cube principal.

[3] Hiral Rathod, Mahendra Singh Sisodia, Sanjay Kumar, Design and Implementation of Image Encryption Algorithm by using Block Based Symmetric Transformation Algorithm (Hyper Image Encryption Algorithm).

[4] Diaa Salama Abd Elminaam, Hatem Mohamed Abdual Kader, and Mohiy Mohamed Hadhoud " Evaluating The Performance of Symmetric Encryption Algorithms" International Journal of Network Security, Vol.10, No.3, PP.213-219, May 2010

[5] Diaa Salama Abdul Minaam, Hatem M. Abdual-Kader, and Mohiy Mohamed Hadhoud "Evaluating the Effects of Symmetric Cryptography Algorithms on Power Consumption for Different Data Types" International Journal of Network Security, Vol.11, No.2, PP.78-87, Sept.

[6] Yan Wang and Ming Hu "Timing evaluation of the known cryptographic algorithms" 2009 International Conference on Computational Intelligence and Security

[7]] ISO, JTC 1/SC 27 (2006). "ISO/IEC 10116:2006 - Information technology --Security techniques -- Modes of operation for an n-bit block cipher". ISO Standards catalogue.

[8] Kuo-Tsang Huang, Jung-Hui Chiu, and Sung-Shiou Shen (2013). "A Novel Structure with Dynamic Operation Mode for Symmetric-Key Block Ciphers" International Journal of Network Security & Its Applications (IJNSA) 1): 19.

[9] NIST Computer Security Division's (CSD) Security Technology Group (STG) (2013). "Current modes". Cryptographic Toolkit. NIST. Retrieved April 12, 2013.

[10] "Stream Cipher Reuse: A Graphic Example". Crypto smith LLC. Retrieved 27 March 2013.

[11] B. Moeller (May 20, 2004), Security of CBC Cipher suites in SSL/TLS: Problems and Countermeasures

