**TITLE OF THE PROJECT**

IMAGE STEGANOGRAPHY

MINI PROJECT REPORT

Submitted in partial fulfilment of the

requirements for the award of the degree of

**BACHELOR OF ENGINEERING**

IN

**INFORMATION TECHNOLOGY**

By

<M.K.Sripriya >< 1602-19-737-018>

<Rehnuma Parveen ><1602-19-737-026>



**Department of Information Technology**

**Vasavi College of Engineering (Autonomous)**

**(Affiliated to Osmania University)**

**Ibrahimbagh, Hyderabad - 31**

**VASAVI COLLEGE OF ENGINEERING(AUTONOMOUS)**

**(AFFILIATED TO OSMANIA UNIVERSITY)**

**HYDERABAD - 500 030**

**Department of Information Technology**

****

**DECLARATION BY CANDIDATE**

We, <M.K.Sripriya> ,<Rehnuma Parveen >, bearing hall ticket number, **<**1602-19-737-018**>**,<1602-19-737-026> hereby declare that the project report entitled **<**” Image Steganography”**>**

Department of Information Technology, Vasavi College of

Engineering, Hyderabad, is submitted in partial fulfillment of the

requirement for the award of the degree of **Bachelor of**

**Engineering** in **Information Technology**

This is a record of bonafide work carried out by me and the

results embodied in this project report has not been submitted to

any other university or institute for the award of any other degree

or diploma.

**<M.K.Sripriya >**

**<1602-19-737-018>**

**<Rehnuma Parveen >**

**<1602-19-737-026>**

ABSTRACT:

Our mini project “Image Steganography” is all about security. Given a secret message that has to passed on to someone, this application will help do that by encoding the given secret message into a selected image. This image can then be sent to the person who the secret message is meant for. The person then decodes the secret message and uses it for whatever it was meant for. Before decoding, the person is also asked for a password which ensures that he is the right person to be decoding the secret message. The entire process is carried out using the lsb algorithm.

INTRODUCTION:

Steganography is the art of hiding the fact that communication is taking place, by hiding information in other information.

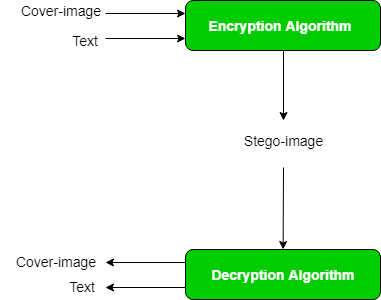
What steganography essentially does is exploit human perception, human senses are not trained to look for files that have information inside of them. The most common use of steganography is to hide a file inside another file.

How is it different from Cryptography?

Steganography is often confused with cryptology because the two are similar in the way that they both are used to protect important information. The difference between two is that steganography involves hiding information so it appears that no information is hidden at all.

Our work:

* Steganography is hiding secret messages in audio, video and image files.
* We choose to implement Image steganography using LSB(least significant bits) algorithm and C language.



To be able to completely implement a steganography project, one must be willing to learn the technique and the many algorithms associated with it. As a team, we have looked at some algorithms to implement this technique of “Image Steganography” and decided to use the lsb(least significant bit) algorithm which is explained in details in the next sections.

Problem we are solving using the image steganography technique: “Image Steganography” is all about security. Given a secret message that has to passed on to someone, this application will help do that by encoding the given secret message into a selected image. This image can then be sent to the person who the secret message is meant for. The person then decodes the secret message and uses it for whatever it was meant for. Before decoding, the person is also asked for a password which ensures that he is the right person to be decoding the secret message.

Now that we know exactly what steganography is and that image steganography is being implemented here, we will now list the requirements of the users to be using our image steganography model.

In an ideal case, there will be two users – the sender and the receiver. The sender having a secret message to pass to the receiver is a requirement. For actually passing the message, the user having a .bmp image and a .txt file containing the message is a requirement. The receiver having the right password for the decoding process is also a requirement for the entire process to go on smoothly.

To summarize, the list of requirements will be:

1. Proper .bmp images
2. Secret message
3. Receiver knowing the correct password
4. Receiver possessing the correct image to decode

TECHNOLOGY

To implement any project successfully, there will be technological requirements which can either be software or hardware requirements.

1. Software requirements:

Since our project was supposed to be based on the C programming language, it is a bare necessity to have the knowledge and syntaxes of the language and a proper compiler and text editor to run and write the programs.

Compiler: Some of the many C compilers include:

1. Borland turbo C
2. Tiny C compiler
3. Portable C compiler
4. GCC compiler
5. Clang

Among the many available compilers, we have installed and used the GCC compiler to run/execute the code for “Image Steganography” that we have written.

Text editor:

To actually write and complete a code in any language, a text editor is important. Some of the famous text editors are:

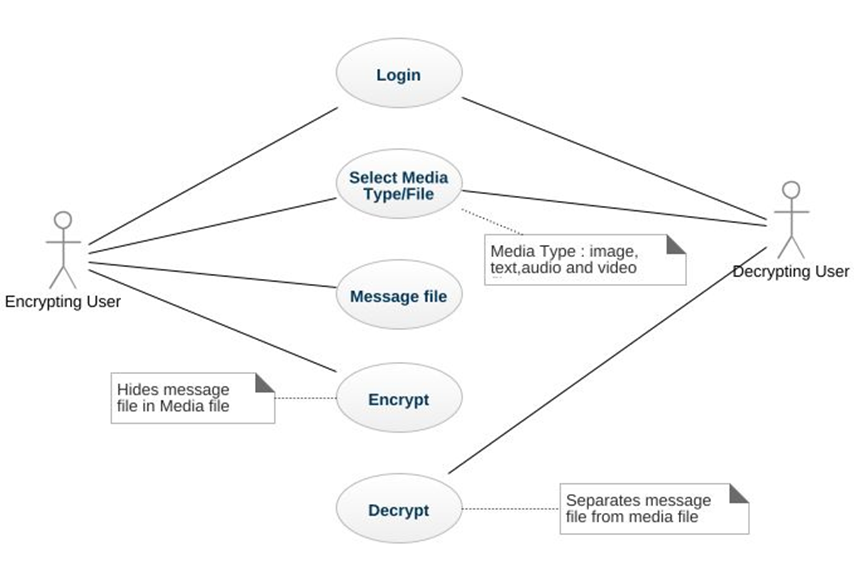
1. Vim editor
2. Notepad
3. Notepad++

Although these are some famous ones, we have installed ad made use of a text editor called Sublime Text. Around 400 lines of the source code for “image steganography” implementation in C has been written there.

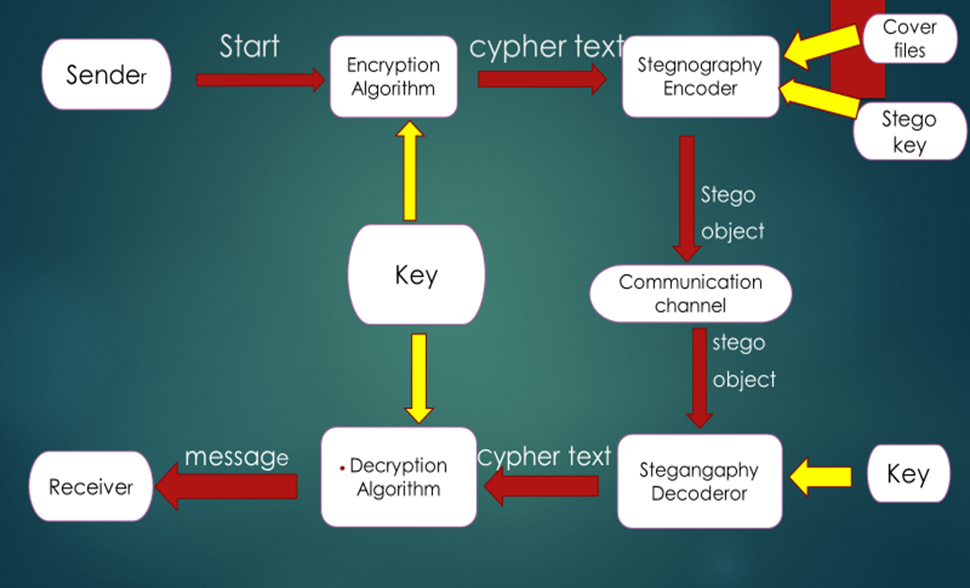
1. Hardware requirements:

Personal computer or a laptop(preferably windows version over linux or mac).

DESIGN:



FLOWCHART



IMPLEMENTATION:

In the current project image steganography is dealt with using data hiding. There are two different methods for image steganography:

1. Spatial methods 2. Transform methods

In spatial method, the most common method used is LSB substitution method.

Least significant bit (LSB) method is a common, simple approach to embedding information in a cover file. In steganography, LSB substitution method is used. I.e. since every image has three components (RGB).

This pixel information is stored in encoded format in one byte. The first bits containing this information for every pixel can be modified to store the hidden text.

For this, the preliminary condition is that the text to be stored has to be smaller or of equal size to the image used to hide the text. LSB based method is a spatial domain method. But this is vulnerable to cropping and noise.

In this method, the MSB (most significant bits) of the message image to be hidden are stored in the LSB (least significant bits) of the image used as the cover image. It is known that the pixels in an image are stored in the form of bits. In a grayscale image, the intensity of each pixel is stored in 8 bits (1byte).

Similarly for a colour (RGB-red, green, blue) image, each pixel requires 24 bits (8bits for each layer). The Human visual system (HVS) cannot detect changes in the colour or intensity of a pixel when the LSB bit is modified.

This is psycho-visual redundancy since this can be used as an advantage to store information in these bits and yet notice no major difference in the image.

**Steps used in LSB steganography**:

1. Steps for hiding message image: 1. Read the image to be used as cover image. Noise is added to make it easier to disguise changes due to embedding the message image. 2. Read the image to be sued as message image. 3. Separate the bit planes of each image.
2. As it is known that the LSB (least significant bit) plane contains the least information associated with any image, and the MSB (most significant bit) plane contains most of the shape, colour information of an image.
3. It is generally ideal to replace up to 4 least bitplanes of the cover image, with the upper 4 bitplanes without revealing changes in the resultant image.
4. Lesser number of bitplanes from the message image could be used, but the retrieved image would become distorted and loses information.
5. 4. Replace the least 4 bitplanes of cover image with the 4 most significant bitplanes from message image. 5. Get the resultant Steganographic image by recombining these bitplanes.
6. b. Retrieving message image: 1. Read the Steganographic image. 2. Extract the required number of bitplanes of the image. 3. Recombining the lower four bitplanes would give thr retrieved message image.

**FORMATION OF STEGO IMAGE**

As soon as we get the new pixel values we form the stego image, which is same as the size of cover image thus the pixel values are placed in the position of the previous values.

Similarly, we take the pixel one by one and then insert the secret image in to them and replace them. Finally, we get the stego image. Stego Image contains the secret image but it cannot be recognized as the secret image.

The change of the pixel value is varied from 0 to 15 which is negligible amount of value. So, pixel values or colors will not be changed in great amount

**ENCODING ALGORITHM**

Input: A grey-level secret image of size (m x n), A grey level Cover Image of size (2m x 2n); Output: Stego Image of size (2m x 2n);

**Steps:**

1.Divide each pixel of the secret image into 4

Parts containing 2 bits.

2.Map these 4 parts into the 4 X-boxes and get the

new values for each part.

3.Insert these values into the LSB position of the

Cover image one by one.

4.End

**IMAGE DECODING**

To decode the secret image in the receiver side we

need to perform the following steps

**GENERATION OF 4 LSB BITS FROM STEGO IMAGE:**

Here we take the pixel values one by one then it istransferred in to the binary values and get the 4LSB bits values from it.

**DECODING ALGORITHM:**

Input: Stego Image of size (2m x 2n); Output: A greylevel Cipher image of size (m x n); Steps:

1.Select each pixel of the Stego-image and take 4

Bits from LSB position of stego image.

2.Then Perform the XOR operation of that 4 bit

LSB and concatenate the four results.

3.Ultimately we get the pixel value of the secret

Image and place one by one to get a secret

Image.

4.End

SOURCE CODE:

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

int get\_image\_data\_offset(FILE\* bmp\_offset) {

fseek(bmp\_offset,10,0);

int offset;

offset=(int)fgetc(bmp\_offset);

return offset;

}

int get\_message\_length(FILE \*fp) {

fseek(fp, 0L, SEEK\_END);

int size = ftell(fp);

fseek(fp, 0L, SEEK\_SET);

return(size);

}

int get\_bit(char the\_byte,int which\_bit) {

return((the\_byte>>8-which\_bit)&1);

}

int main(int argc,char\*\* argv) {

unsigned char mask\_table[] = { 0x01, 0x02, 0x04, 0x08, 0x10, 0x20, 0x40, 0x80 };

FILE \*file\_handle;

FILE \*message\_handle;

FILE \*hidden\_message\_handle;

if(argc!=5) {

printf("Please give correct command line arguments for encoding or decoding :\n\n");

printf("\*\*\* Steganography by LSB substitution\*\*\*\nUsage: %s [-e][-d] <source file> <destination file> <text file>\n-e : Add text to image\n -d : Get text from image\n",argv[0]);

exit(1);

}

int mode;

if(!strcmp(argv[1],"-e"))

mode=1;

else if(!strcmp(argv[1],"-d"))

mode=0;

else {

printf("Please give correct command line arguments for encoding or decoding :\n\n");

printf("\*\*\* Steganography by LSB substitution\*\*\*\n\nUsage: %s <mode> <source file> <destination file> <text file>\nMode - e = encrypt \n d= decrypt \n",argv[0]);

exit(1);

}

file\_handle=fopen(argv[2],"r");

if (file\_handle == NULL) {

fprintf(stderr, "Can't open input file %s\n",argv[2]);

exit(1);

}

hidden\_message\_handle=fopen(argv[3],"w");

if (hidden\_message\_handle== NULL) {

fprintf(stderr, "Cannot create output file %s\n",argv[3]);

exit(1);

}

int c=0;

char tmp\_sig\_cpy;

int offset=get\_image\_data\_offset(file\_handle);

rewind(file\_handle);

for(int i=0;i<offset;i++) {

tmp\_sig\_cpy=fgetc(file\_handle);

fputc(tmp\_sig\_cpy,hidden\_message\_handle);

c++;

}

char file\_buffer;

char message\_buffer;

if(mode) {

printf("\nYou have successfully encoded your secret message.\n\n Please send the image with your hidden message to the relevant person who will be asked for the password.\n");

message\_handle=fopen(argv[4],"r");

if (message\_handle== NULL) {

fprintf(stderr, "Can't open text input file %s\n",argv[4]);

exit(1);

}

int hidden\_message\_length=get\_message\_length(message\_handle);

fputc(hidden\_message\_length,hidden\_message\_handle);

c++;

do {

int bit\_of\_message;

if(!feof(message\_handle)) {

message\_buffer=fgetc(message\_handle);

for(int i=1;i<=8;i++) {

file\_buffer=fgetc(file\_handle);

c++;

int file\_byte\_lsb = file\_buffer & 1;

bit\_of\_message=get\_bit(message\_buffer,i);

if(file\_byte\_lsb==bit\_of\_message) {

fputc(file\_buffer,hidden\_message\_handle);

}

else {

if(file\_byte\_lsb==0)

file\_buffer = (file\_buffer | 1);

else

file\_buffer = (file\_buffer & ~1);

fputc(file\_buffer,hidden\_message\_handle);

}

}

}

else {

tmp\_sig\_cpy=fgetc(file\_handle);

fputc(tmp\_sig\_cpy,hidden\_message\_handle);

c++;

}

} while(!feof(file\_handle));

fclose(message\_handle);

}

else {

char password[] = "teamB24";

char pass[100];

int z = 0;

char c;

printf("If you are the right person to receive the secret message, then enter the password : \n");

//scanf("%s",pwd);

//printf("\n\n\t\t\t\t ENTER PASSWORD: ");

//printf("\n\n\t\t\t\t ENTER PASSWORD: ");

while((c=getch())!=13)

{

pass[z++]=c;

printf("%c",'\*');

}

pass[z]='\0';

if(strcmp(password,pass)){

printf("\n\nYou have entered the wrong password. Please try the decoding process again.\n");

exit(1);

}

printf("Successfully decoded!!\n\n open the file to view the secret message you were supposed to receive.\n");

message\_handle=fopen(argv[4],"w");

if (message\_handle== NULL) {

fprintf(stderr, "Can't open text input file %s\n",argv[4]);

exit(1);

}

int message\_length=fgetc(file\_handle);

for(int i=0;i<message\_length;i++) {

char temp\_ch='\0';

for( int j=0;j<8;j++) {

temp\_ch=temp\_ch<<1;

file\_buffer=fgetc(file\_handle);

int file\_byte\_lsb = file\_buffer & 1;

temp\_ch|=file\_byte\_lsb;

}

fputc(temp\_ch,message\_handle);

}

fclose(message\_handle);

}

fclose(file\_handle);

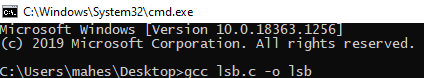
fclose(hidden\_message\_handle);

}

TESTING:

To compile and execute the above source code for “Image Steganography”, the following are the commands:

Compilation: gcc lsb.c -o lsb



To run the code to get the output or either encoding or decoding:

$ ./lsb

You will be presented with a message as shown below:

Please give correct command line arguments for encoding or decoding :

\*\*\* Steganography by LSB substitution\*\*\*

Usage: ./lsb [-e][-d] [source file] [destination file] [text file]

Mode

-e : Add text to Image

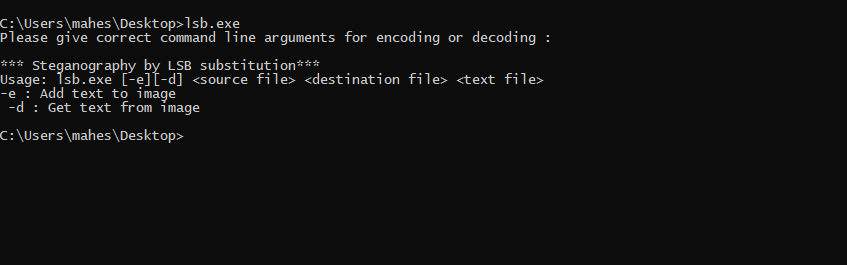
-d : Extract text from Image

To create a stegano image from a plain image and a text file with the message. Run the program as below

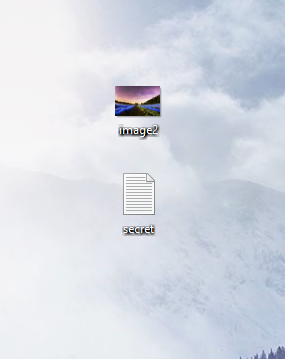
$ ./lsb -e sample.bmp stegano\_sample.bmp message.txt

To reterive the text that was put into an image with the above program. Run the program as shown below

$ ./lsb -d stegano\_sample.bmp intermediate.bmp original\_message.txt



The prerequisites for running the encoding command are the existence of a text file containing the secret message and a bmp image to hide the message in.



In the absence of these, the code sends a message saying that the file was not found.

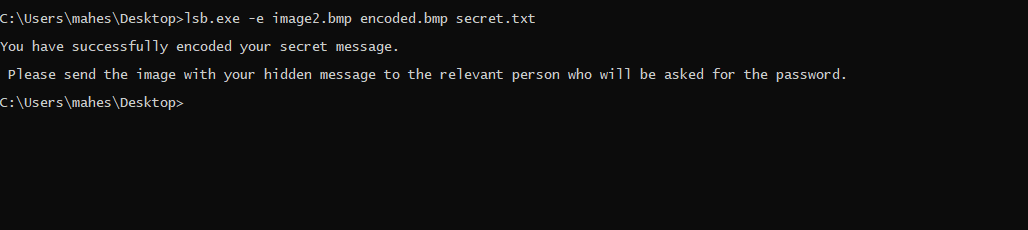
For the decoding process, although there are no files required beforehand (other than the encoded bmp image), the person decoding the text from image is required to have the knowledge of a password that starts the decoding process, without which the program gets terminated.

RESULTS:

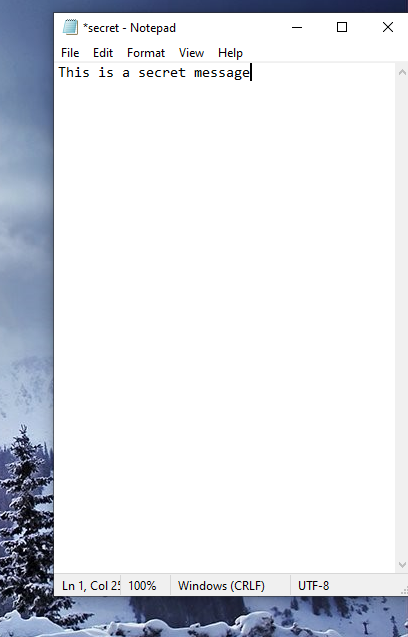
Encoding process:

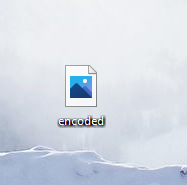
The first step is to have the files required, i.e “secret.txt” and “image2.bmp” here.

Then we just have to give the correct command for running with appropriate command line arguments.



This is the message received by the user after the end of the encoding process. The message is secret.txt has been encoded into image2.bmp and the final output is an encoded image encoded.bmp.



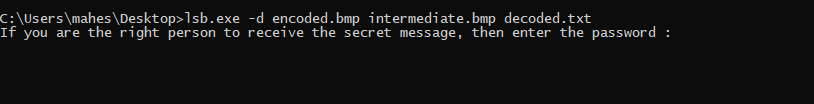


Encoded image created on desktop.

The final step in this encoding process is to send this encoded image “encoded.txt” to any person who is meant to receive the secret message in “secret.txt” and make her/him aware of the password for the decoding process.

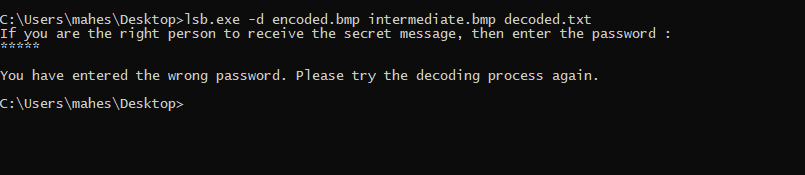
Decoding process:

After making sure that we have the correct “encoded.bmp” encoded image file, the next step is to run the command for decoding with correct command line arguments.



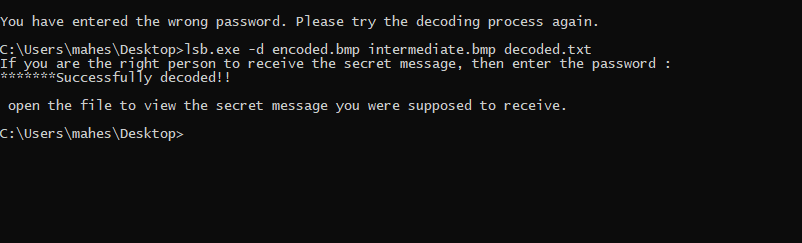
The user is then being asked to enter the password.

If the password entered is wrong:



The user will then have to try again.

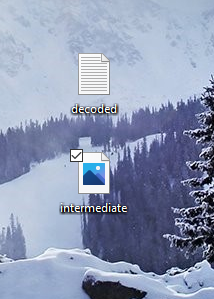
If the entered password is correct:



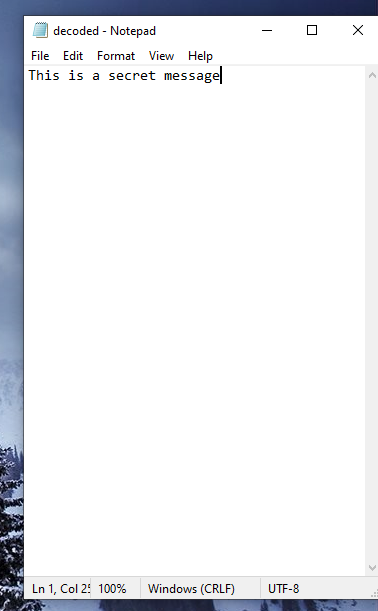
The secret message was successfully put into a text file called “decoded.txt”.

“intermediate.bmp” is an image file being used in the intermediate step of the decoding process.

Both these files are now available on the desktop.



By opening the “decoded.txt” file, the user can now see the secret message that was originally in “secret.txt”.



This completes the decoding process.

GITHUB REPOSITORY:

<https://github.com/sripriyamaturi/mini-project>

The above mini project repository consists of the abstract, design document, source code and the report for “Image steganography”.

ADDITIONAL LEARNINGS:

Apart from the immense understanding and practice of C language and “Image steganography we obtained throughout the course of this project, there were many additional learnings too:

We have learned to work in a team. Each of us have different schedules but had the same goal which made us work towards it together and hence we completed it together. Doing the entire project during these uncertain times of covid was also a challenge. We spent time to coordinate through Microsoft teams and had our discussions.

We have learned to meet deadlines. Throughout these months, we successfully completed and submitted the mini project abstract, the design document, the different modules in the project and not the report too.

Thinking out of the box. We were constantly thinking about how to make this project better than the existing ones and user friendly.

DISCUSSION AND FUTURE WORK:

Steganography is an old art which has been in practice since time unknown.

This research has opened new avenues for us and scores of new ideas have now sprouted. of these some are blurred but many pose a clear picture of our future directions. Some of these are outlined below:



 Four main factors, viz. hiding capacity, perceptual transparency, robustness and tamper resistance, were identified elsewhere to characterize the data hiding techniques in steganography. We mainly concentrated on the hiding capacity. In our technique, it is very difficult to detect or suspect the presence of the secret message because a very little change occurs which cannot be detected by human eye. So no one can suspect the presence of message.



 But if suspected then the message can be decoded. So, we have to apply some more algorithms for the security purpose. The LSB algorithm can also be modified in various ways to increase the security of the data.



 By tempering the coded image, our secret data may be lost. Tempering can be cropping, blurring of the image or resizing the image. So this operation may lead to the loss of data. So some more work to be done on these so that the stego image will be temper resistant.



 As far as perceptual transparency is concerned it was beyond the scope of this thesis. Future work should take this into account. This aspect promises greater room for research. In this context Steganography is an old art which has been in practice since time unknown.

This research has opened new avenues for us and scores of new ideas have now sprouted. of these some are blurred but many pose a clear picture of our future directions. Some of these are outlined below:



 Four main factors, viz. hiding capacity, perceptual transparency, robustness and tamper resistance, were identified elsewhere to characterize the data hiding techniques in steganography. We mainly concentrated on the hiding capacity. In our technique, it is very difficult to detect or suspect the presence of the secret message because a very little change occurs which cannot be detected by human eye. So no one can suspect the presence of message.



 But if suspected then the message can be decoded. So we have to apply some more algorithms for the security purpose. The LSB algorithm can also be modified in various ways to increase the security of the data.



 By tempering the coded image, our secret data may be lost.Tempering can be cropping, blurring of the image or resizing the image. So this operation may lead to the loss of data. So some more work to be done on these so that the stego image will be temper resistant.



 As far as perceptual transparency is concerned it was beyond the scope of this thesis. Future work should take this into account. This aspect promises greater room for research in this context

REFERENCES:

1. <https://towardsdatascience.com/steganography-hiding-an-image-inside-another-77ca66b2acb1>
2. <https://github.com/bjchau/Steganography>
3. <https://www.ijltet.org/wp-content/uploads/2015/02/60.pdf>
4. <https://www.geeksforgeeks.org/image-steganography-in-cryptography/>
5. <https://www.programmingassignmenthelper.com/steganography-hide-and-display-data-inside-images/>