A Project Report on

'BPCS Algorithm based Steganography with Integration of RSA'

Submitted to: Smart Internz

Submitted by :
Sarjak Devkota 20BCE2922 (VIT Vellore)
Muhammed Irfan Kuzhylangattil 20BCI0200 (VIT Vellore)
Sarthak Verma 20BCI0324 (VIT Vellore)
Sanjith kumar 20BCE7156 (VIT Amravati)

ACKNOWLEDGEMENT

We would like to thank our professor who has been the great inspiration and who has provided sufficient background knowledge and understanding of this subject.

Our humble prostration goes to you for providing all the necessary resources and environment, which have aided us to complete this project successfully. We owe a very special thanks to you for being a great source of support and encouragement. We feel privileged to extend our deep sense of gratitude to our parents for their support and encouragement.

PREFACE

This report is about our combined project based on Cryptography and Steganography. Steganography alone is not secured, because once an attacker comes to know that there is a data hidden inside a particular vessel, then decoding is quite easy. Encryption is secured in its own way, because even if we know that the given content is encrypted on the basis of certain algorithm, even then we can't find the original message because the complexity in breaking the algorithms required complex systems.

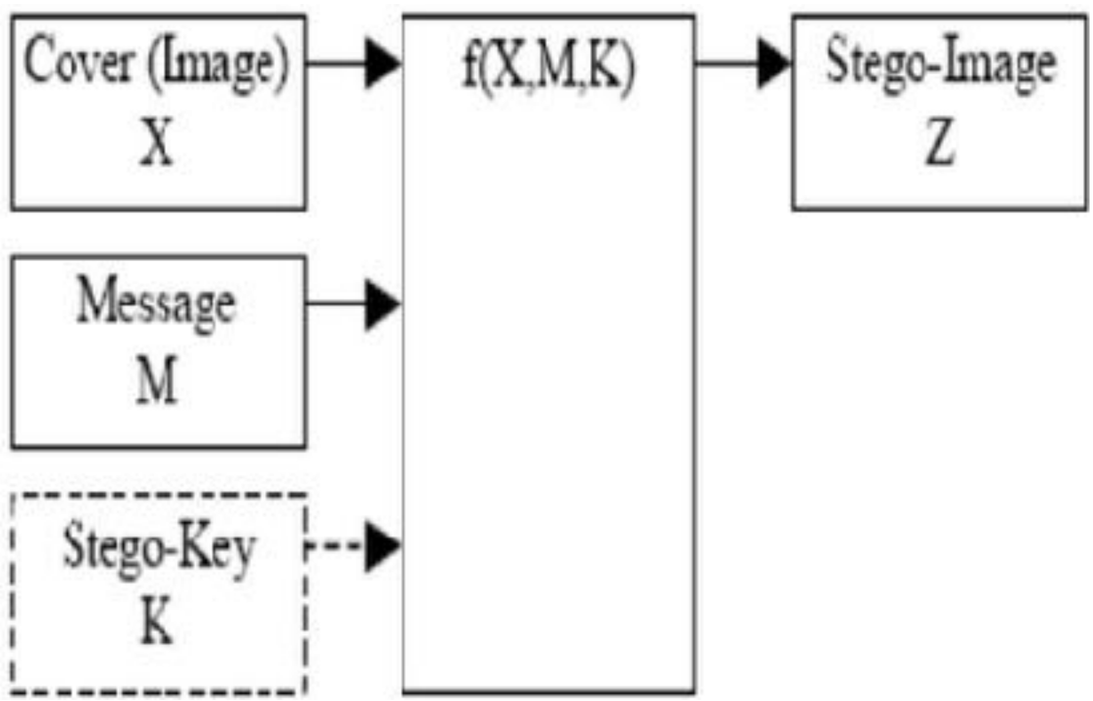
The core idea about the project is implementing steganography with encryption together and integrating to create a secured channel. This report also contains the comparison of BPCS algorithm with another existing algorithm like LSB.

S. No	Topic	Page No.
1.	Introduction	4
2.	Objective of The Project Literature Review	12
3.	Innovation component in the Project	15
4.	Work done and implementation	16
A.	Methodology Adapted	20
B.	Dataset used	
C.	Tools used	39
D.	Screenshot, Demo and Visualization	43
5.	Results and discussion with Visualization	45
6.	References	46

Introduction

Cryptography and steganography are cousins in the spy craft family: the former scrambles a message so it cannot be understood, the latter hides the message so it cannot be seen. A cipher message, for instance, might arouse suspicion on the part of the recipient while an invisible message created with steganographic methods will not

A basic steganographic model is shown in Figure 1. The message ‘M’ is the secret data that the Sender wishes to hide without any suspicion. The secret data can be audio, video, image, text. The cover ‘X’ is the original image, audio file, video file, in which the secret message ‘M’ is to be embedded. The cover ‘X’ is also called as “Message Wrapper”. It is not necessary that the cover ‘X’ and the message ‘M’ should have homogeneous structure. For example, text message or an audio file can also be hidden into video or image. In this project the cover ‘X’ is image and Message ‘M’ is text.



The goal of steganography is to hide a message in plain sight. BPCS is a method to embed a message in an image by replacing all "complex" blocks of pixels in the image with portions of our message. It turns out that portions of the image with high complexity can be entirely removed (or in this case, replaced with our message) without changing the appearance of the image at all. Because most blocks of pixels are complex (i.e., with complexity above some threshold, alpha), you can usually replace around 45% of an image with a hidden message. Below, the 300x300 image on the right contains the text of an entire novel, while still looking virtually identical to the vessel image on the left.

VESSEL IMAGE

ENTIRE NOVEL EMBEDED IN THIS IMAGE

Note that with BPCS, the hidden message doesn't have to be text. It can be any file type, including another image.

You could upload a profile photo to a website that contains a secret image. Or you could embed an image of a turtle inside an image of a turtle inside an image...turtles all the way down.

This is an implementation of the method discussed in: Kawaguchi, Eiji, and Richard O. Eason. "Principles and applications of BPCS steganography." In Photonics East (ISAM, VVDC, IEMB), pp. 464-473. International Society for Optics and Photonics, 1999.

The goal of steganography is to hide things in plain sight. For this reason, BPCS doesn't use a secret key or password for encoding and decoding. However, aside from varying the alpha parameter, one way to customize the BPCS procedure is by adding custom encryption and decryption to the message before and after using BPCS and we have done this with RSA algorithm. There are various ways the complexity regions are described in various research papers, but in this project, we use complexity definition on the basis of alpha parameter.

The important step in BPCS steganography is to find “complex” region in the vessel image so that data from secret image can be hidden without any suspicion. Also, there is no standard definition of complexity. There are basically three methods of complexity measure. However, in our experiment and in this paper, we focus on complexity measure based on length of black and white border in binary image. The total length of black and white border is equal to the summation of the number of color changes along the rows and columns in an image

$$\alpha = \frac{k}{2 \times 2^m \times (2^m - 1)}$$

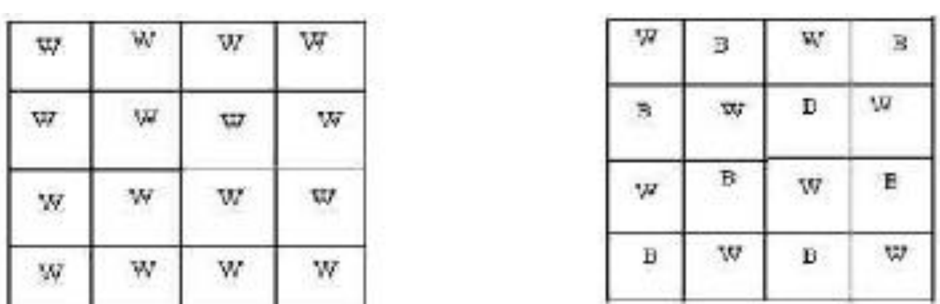


Figure 4: (a) all white pixels in image (b) black-white checker board

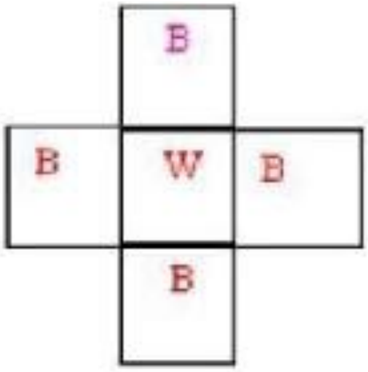


Figure 3: single white pixel surrounded by four black pixels

5. Proposed BPCS Steganography Algorithm

a) Consider a color image as vessel image. Make the size of image as 512 x 512.

b) Convert the vessel image to gray scale image.

c) Consider a gray scale secret image and make the size of image as 256 x 256.

d) Convert the vessel image and the secret image which are in pure binary Code (PBC) form into Canonical Gray Code (CGC) form.

e) Perform bit plane slicing on vessel image as well as on secret image.

f) Calculate complexity measure ‘alpha’ (α) for each block of each bit plane of vessel image.

g) Calculate α for each block of each bit plane of secret image.

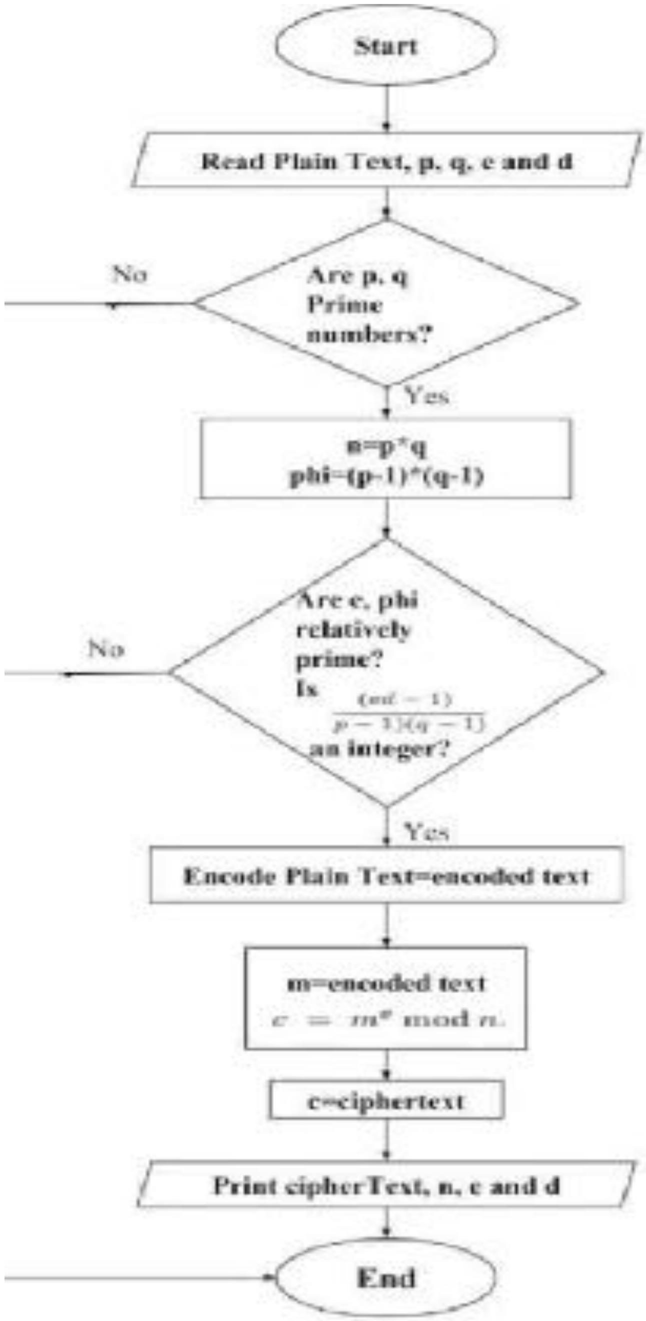
h) Perform conjugation operation on the ‘simple’ or ‘informative’ blocks of the secret image.

I) Perform embedding operation to embed secret image in vessel image.

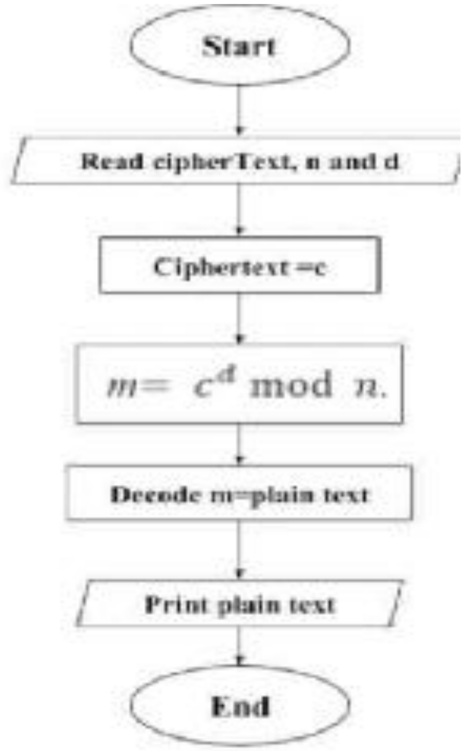
j) Convert the CGC form embedded image to PBC image.

Steganography followed by Encryption using RSA

Cryptography and steganography are cousins in the spy craft family: the former scrambles a message so it cannot be understood, the latter hides the message so it cannot be seen. A cipher message, for instance, might arouse suspicion on the part of the recipient while an invisible message created with steganographic methods will not. In fact, steganography can be useful when the use of cryptography is forbidden: where cryptography and strong encryption are outlawed, steganography can circumvent such policies to pass message covertly. However, steganography and cryptography differ in the way they are evaluated: steganography fails when the” enemy” is able to access the content of the cipher message, while cryptography fails when the” enemy” detects that there is a secret message present in the steganographic medium (Johnson and Jajodia, 1998).



(a) encryption



(b) decryption

OBJECTIVE

In today's information age, information sharing and transfer has increased exponentially. The threat of an intruder accessing the secret information has been an ever-existing concern for data communication experts. Cryptography and steganography are the most widely used techniques to overcome this threat. This paper is based on hybrid cryptographic techniques based on RSA algorithms to achieve data encryption and compression technique to store large amount of data. A combination of both provides superior security control. The suggested algorithm is modified BPCS (Bit Plane Complexity Segmentation) steganography technique that can replace all the “noise-like” regions in all the bit-planes of the cover image with secret data without deteriorating the image quality. According to the experiments, the messages can be successfully camouflaged in the cover image, and the stego images have satisfactory quality. Moreover, our scheme allows for a large capacity of embedded secret data and can be extracted from stego-image without the assistance of original image.

Today's information world is a digital world. Data transmission over an unsecure channel is becoming a major issue of concern nowadays. And at the same time intruders are spreading over the internet and being very active. So, to protect the secret data from theft some security measures need to be taken. In order to keep the data secret various techniques have been implemented to encrypt and decrypt the secret data. Cryptography and Steganography are the two most prominent techniques from them. But these two techniques alone can't do work as much efficiently as they do together. Steganography is a Greek word which is made up of two words Stegano and graphy. Stegano means hidden and graphy means writing i.e., Steganography means hidden writing. Steganography is a way to hide the fact that data communication is taking place. While cryptography converts the secret message in other than human readable form but this technique is having a limitation that the encrypted message is visible to everyone. In this way over the internet, intruders may try to apply heat and trial method to get the secret message. Steganography overcome the limitation of cryptography by hiding the fact that some transmission is taking place. In steganography the secret message is hidden in other than original media such as Text, Image, video and audio form. These two techniques are different and having their own significance. So, in this paper we are going to discuss various cryptographic and steganographic techniques used in order the keep the message secret. Thus, our project aims to depict steganography (using modifies BPCS for high data embedding and Cryptography using RSA algorithm.

LITERATURE REVIEWS

SI NO.	TITLE, AUTHOR, JOURNAL	OBJECTIVE	METHODOLOGY	LIMITATIONS / CONCLUSION
1.	Review: Steganography – Bit Plane Complexity Segmentation (BPCS) Technique SHRIKANT S. KHAIRE International Journal of Engineering Science and Technology Vol. 2(9), 2010, 4860-4868	Description of BPCS steganographic model, comparison with other models. Future prospects and to describe edge bordering as new way of complex region determiner.	A mathematical formulation for analysis of BPCS steganographic techniques is introduced. Novel and rigorous approach is adapted to arrive at stenographic capacity of BPCS based image, data hiding capacity and the ways to achieve that is discussed on the basis of alpha parameter(complex region parameter)	The challenge of using other algorithms is limitation of image embedding in the vessel(10-20 percent) while in case of BPCS almost 50 percent because all the bit planes including the MSB bit plane can be encoded .
2.	Hosam, O., & Ahmad, M. H. (2019). Hybrid design for cloud data security using combination of AES, ECC and LSB steganography. International Journal of Computational Science and Engineering, 19(2), 153-161.	To develop a hybrid solution to tackle the key management problem. To achievestrong security posture and efficient key management and distribution for multiple	In this paper a hybrid design for ensuring data security in cloud and resolve the key management problem is proposed. Using an intelligent mix of Steganography, Elliptic Curve Cryptography (ECC)	The mechanism presented in this paper allows the encryption of multiple files with their own respective AES keys. This reduces the chances of traffic or side channel analysis to determine

		users using Steganography, ECC and AES achieve	and Advanced Encryption Standard (AES. AES to encrypt the large blocks of data of any size and then encrypt the symmetric key with public key of any sharing partner is used. The encrypted symmetric key is also stored in an image using steganography	the key.

SI NO.	**TITLE, AUTHOR, JOURNAL**	**OBJECTIVE**	**METHODOLOGY**	**LIMITATIONS / CONCLUSION**
3.	2001, October). Analysis of LSB based image steganography techniques. In *Proceedings 2001 International Conference on Image Processing (Cat. No. 01CH37205)* (Vol. 3, pp. 1019-1022). IEEE	To classify various image steganography techniques alongwith overview, importance and challenges to steganography technique	A mathematical formulation for analysis of LSB based steganographic techniques is introduced. Novel and rogourous approach is adapted to arrive at stenographic capacity of LSB based image data hiding techniques. Capacity is defined in terms of detectability.	The technical challenge of data hiding is finding redundant bits in carrier signal that cannot be statistical and perceptually attacked [. Uncompressed file formats (BMP, GIFF, TIFF) based on lossless compression provides high data capacity and are more convenient for data hiding algorithm
4.	Hosam, O., & Ahmad, M. H.	To develop a hybrid	In this paper a hybrid	The mechanism

	(2019). Hybrid design for cloud data security using combination of AES, ECC and LSB steganography. *International Journal of Computational Science and Engineering*, *19*(2), 153-161.	solution to tackle the key management problem. To achievestrong security posture and efficient key management and distribution for multiple users using Steganography, ECC and AES achieve	design for ensuring data security in cloud and resolve the key management problem is proposed. Using an intelligent mix of Steganography, Elliptic Curve Cryptography (ECC) and Advanced Encryption Standard (AES. AES to encrypt the large blocks of data of any size and then encrypt the symmetric key with public key of any sharing partner is used. The encrypted symmetric key is also stored in an image using steganography	presented in this paper allows the encryption of multiple files with their own respective AES keys. This reduces the chances of traffic or side channel analysis to determine the key.

SI NO.	TITLE, AUTHOR, JOURNAL	OBJECTIVE	METHODOLOGY	LIMITATIONS / CONCLUSION
5.	Sanchez, A., Conci, A., Zelikovic, E., Behlilovic, N., & Karahodzic, V.(2012, October). A new approach to relatively short message steganography. In 2012 IX International Symposium on Telecommunications (BIHTEL) (pp.1-4). IEEE.	Provide the study on increasing range of data types which are swapped over this network (video, audio, text messages..), highlight the security problem that this way of communication	Using GA algorithm (Genetic Algorithm) or a combination of GA and PR (Path Relinking) algorithms, the level of difference between the original and stego image is significantly reduced.	By using GA the alterations between the original image and the image surrounded with secret data can be reduced. However, the alteration between the original image and the image with embedded evidence still remains, while the achieved developments are paid with an increase of computational complexity.
6.	Khari, M., Garg, A. K., Gandomi, A. H., Gupta, R., Patan, R., & Balusamy, B. (2019). Securing data in Internet of Things	The paper aims to implement entiliptic Variety Galois cryptography (EGC)	The elliptic Variety Galis cryptography protocol is implemented. A cryptography technique is employed to encode confidential knowledge that came	Some official videos, such as MPEG 2 and MPEG 4 help to achieve a high compression ratio. Although high-

SI NO.	TITLE, AUTHOR, JOURNAL	OBJECTIVE	METHODOLOGY	LIMITATIONS / CONCLUSION
	(IoT) using cryptography and steganography techniques. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 50(1), 73-80.	protocol for protection against data infiltration during transmission over the IoT network.	from different medical sources. A Matrix XOR coding Steganography technique is employed to embed the encrypted knowledge into an occasional complexes image.	performance efficiency is achieved, there is a loss in the compression ratio
7.	Sharma, M. H., Mithlesh Arya, M., & Goyal, M. D. (2013). Secure image hiding algorithm using cryptography and steganography. IOSR Journal of Computer Engineering (IOSR-JCE) e-ISSN,2278-0661.	Securing image by using combination of cryptography and steganography.	In this paper least significant bit (LSB) is used to hide hidden image into video, which provides the new dimensions to the image steganography. This project worked on two major techniques of data security i.e. Cryptography and Steganography. In the given system these two techniques provides higher security to our data	
8.	Abood, M. H. (2017, March). An	To propose a algorithm that	Cryptography and steganography are used to ensure security of	The proposed work cannot

	efficient image cryptography using hash-LSB steganography with RC4 and pixel shuffling encryption algorithms. In 2017 Annual Conference on New Trends in Information & Communications Technology Applications (NTICT) (pp. 86-90). IEEE.	ensures the encryption and decryption using RC4 stream cipher and GB pixel shuffling with steganography by using hash-least significant Bit (HLSB) that make use of hash function to developed significant way to insert data bits in LSB bits of RGB pixels of cover image	transmitted data. RC4 and pixel shuffling encryption algorithm is used to encrypt the secret image and Hash-LSB is embedded encrypted image into the selected least significant bits of GB image and then sent in receiver side the image is reconstructed from stego GB image and use RC4 and pixel shuffling decryption algorithms to obtain the original image. In this paper trying to verify the confidentiality of grayscale image that makes uses of pixel shuffling and RC4 stream cipher for cryptography and Hash-LSB for steganography.	hide the colour image within other colour image, it can only hide grayscale image inside the RGB image.

INNOVATION COMPONENET OF THE PROJECT

Our project demonstrates the combination of RSA with BPCS steganography. Among the resources we found in the internet, we either found only the steganography or RSA, but the secured channel is when both the components are intertwined. Thus, in this project we bring the implementation using the different modules of Python, the image steganography with BPCS steganography protocol. We even realized the disadvantages of other algorithms posed, like LSB where only $1/8^{th}$ of the image can be embedded, because LSB uses only the last 4 significant bits for steganography. But in case of RSA, the use of bits for replacement are based on the complexity region. In this project we have implemented the complexity region based on the theorem we have mentioned earlier in the document. Nowhere we could find the project intertwining the cousins of security mechanism: the cryptography and the steganography. We have implemented the BPCS steganography model using the python language and its modules while the RSA were implemented using C sharp. Thus, the innovation stands in hiding data upto 50 percent of the capacity of image and also encrypting it with one of the most secured public key encryptions algorithms, the RSA. We were just settling with Steganography with BPCS algorithm, and we had completed this project in the mid of November, but one of our team members sought an idea of further extending the project so that the encryption portion could be depicted too along with the steganography. Thus, from embedding to encryption, our model certainly is very much secured in the recent time.

Work and Implementation

We collectively sought for the comparison of BPCS steganography model with other model like LSB's and we started searching and collecting the data regarding the efficiency of both the model.

The image with LSB steganography model

Image with BPCS model

We examined around 40-50 images with LSB and BPCS algorithm

Steganography image with LSB

Steganography using BPCS

With LSB steganography certainly the image seems to have some noisy regions and also with distortion, what we also came to know was LSB steganography is well implemented with .png format and with .jpg/.jpeg format the distortion seems to be noticeable. Thus, for large embedding (nearly the text of a novel) can be well implemented using the BPCS steganography. Using the internet readymade tool for LSB, we compared the embedding capacity and the limitation of embedding with both BPCS and LSB. Our project did not even throw exception for an entire cryptography book, but the internet tool for LSB showed exception of file size with only few MB’s. Thus, we came to understand from various experimentation that the LSB steganography is not any near when it comes to BPCS steganography.

We took the reference of few research papers and few implementations of steganography and RSA encryption. We certainly had to go through immense research papers and large number of projects, whose analysis we have mentioned in the review section. But we had our own idea of channeling the cryptography after steganography and our combination of BPCS with RSA, is certainly unique and poses lots of security benefits. Our project uses the unique combination of RSA and BPCs.

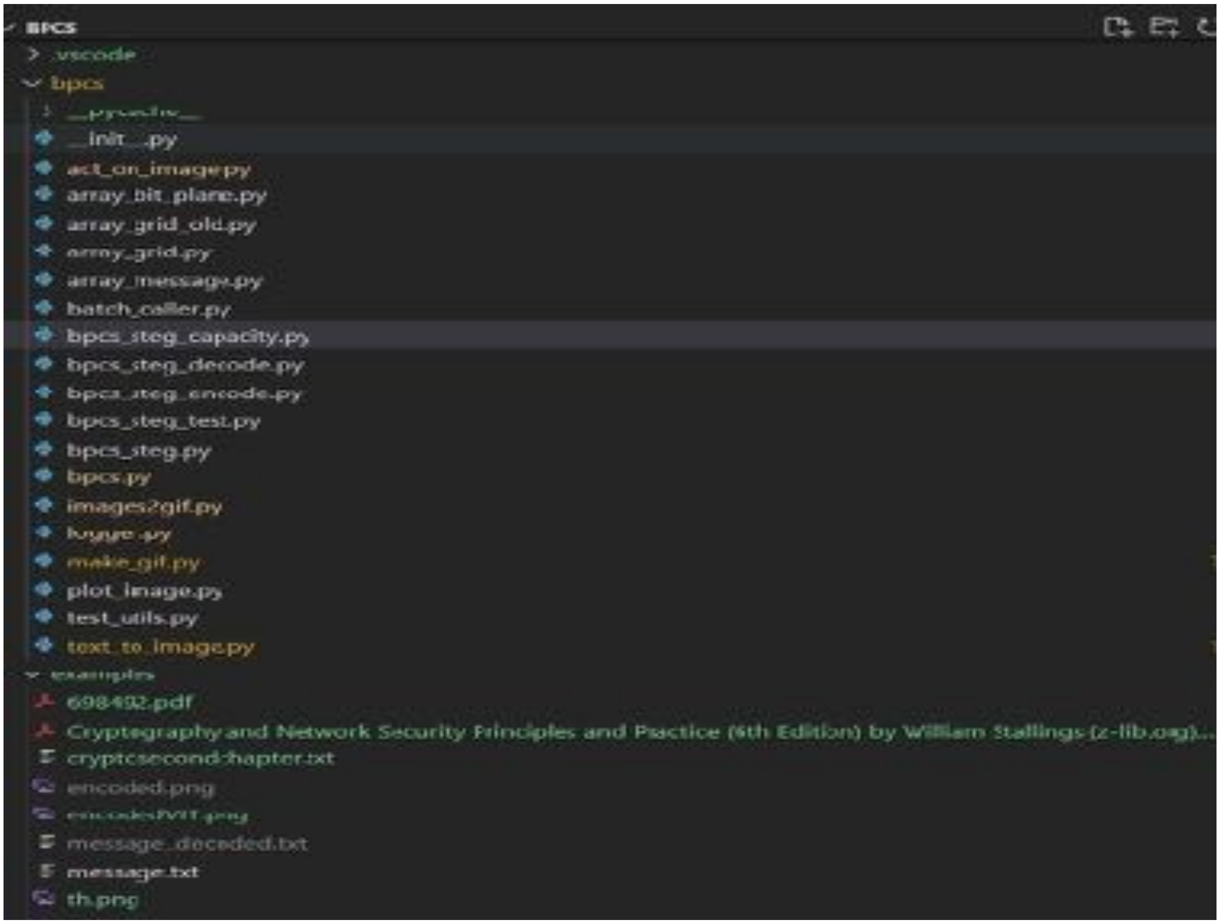
IMPLEMENTATION OF BPCS ALGORITHM

Language Used: Python

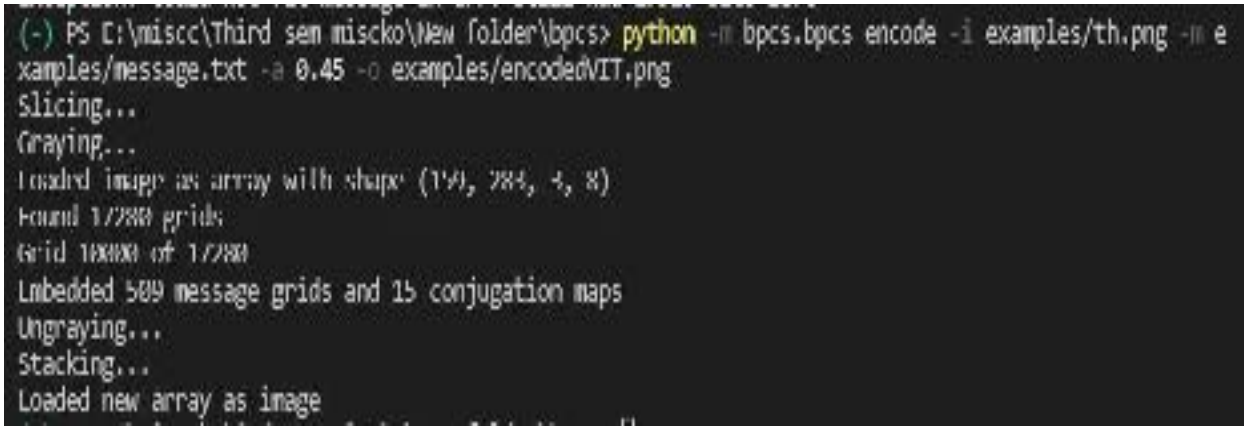
IDE USED FOR THE PROJECT IS VS CODE

- Libraries of Python imported:
    1. NumPy
    2. FUNCTOOLS
    3. PILLOW
    4. MATH
    5. OS
    5. LOGGER and Matplotlib

ALL The python files are kept in BPCS directory and the Image to Be encoded, Decoded and text to be hidden all are present in the same directory



PYTHON FILES FOR VARIOUS CODE SECTIONS



ENCODING PROCEDURE

VESSEL IMAGE

ENCODED IMAGE

Vessel Image looks the same as encoded message, but if we zoom, we can notice the distortion to some extent because we have used significantly larger text size inside a small vessel size.

```
PS C:\Users\Legion\Downloads\bpcs-master (1)\bpcs-master> python -m bpcs.bpcs decode -i examples/encoded.png -a 0.45 -o examples/message_decoded.txt
Slicing...
Graying...
Loaded image as array with shape (300, 300, 3, 8)
Found 34656 grids
Grid 10000 of 34656
Grid 20000 of 34656
Grid 30000 of 34656
Found 531 out of 34656 grids with complexity above 0.45
Found 516 message grids and 15 conjugation maps
PS C:\Users\Legion\Downloads\bpcs-master (1)\bpcs-master>
```

<u>Decoding procedure</u>

```
PS C:\Users\Legion\Downloads\bpcs-master (1)\bpcs-master> python -m bpcs.bpcs capacity -i examples/vessel.png -a 0.45
Slicing...
Graying...
Loaded image as array with shape (300, 300, 3, 8)
Creating histograms of image complexity...
Found 34656 grids
C:\Users\Legion\Downloads\bpcs-master (1)\bpcs-master\bpcs\bpcs_steg_capacity.py:15: FutureWarning: Using a non-tuple sequence for multidimensional indexing is deprecated; use `arr[tuple(seq)]` instead of `arr[seq]`. In the future this will be interpreted as an array index, `arr[np.array(seq)]`, which will result either in an error or a different result.
  vals = [arr_bpcs_complexity(arr[dims]) for dims in get_next_grid_dims(arr, grid_size)]
Grid 10000 of 34656
Grid 20000 of 34656
Grid 30000 of 34656
8479.0 of 34656.0 grids available with alpha of 0.45
Approximately 66784.0 bytes of storage space can fit in this vessel image.
66784.0 byte message would utilize 42.3% of the vessel image.
PS C:\Users\Legion\Downloads\bpcs-master (1)\bpcs-master>
```

<u>Checking the vessel image capacity (here, 42.3%)</u>

GUI AND COMMAND LINE UTILITES OF OUR PROJECT

```
$ python -m bpcs.bpcs encode -i
examples/vessel.png -m examples/message.txt -a
0.45 -o examples/encoded.png
```

Command line Utility for Encoding

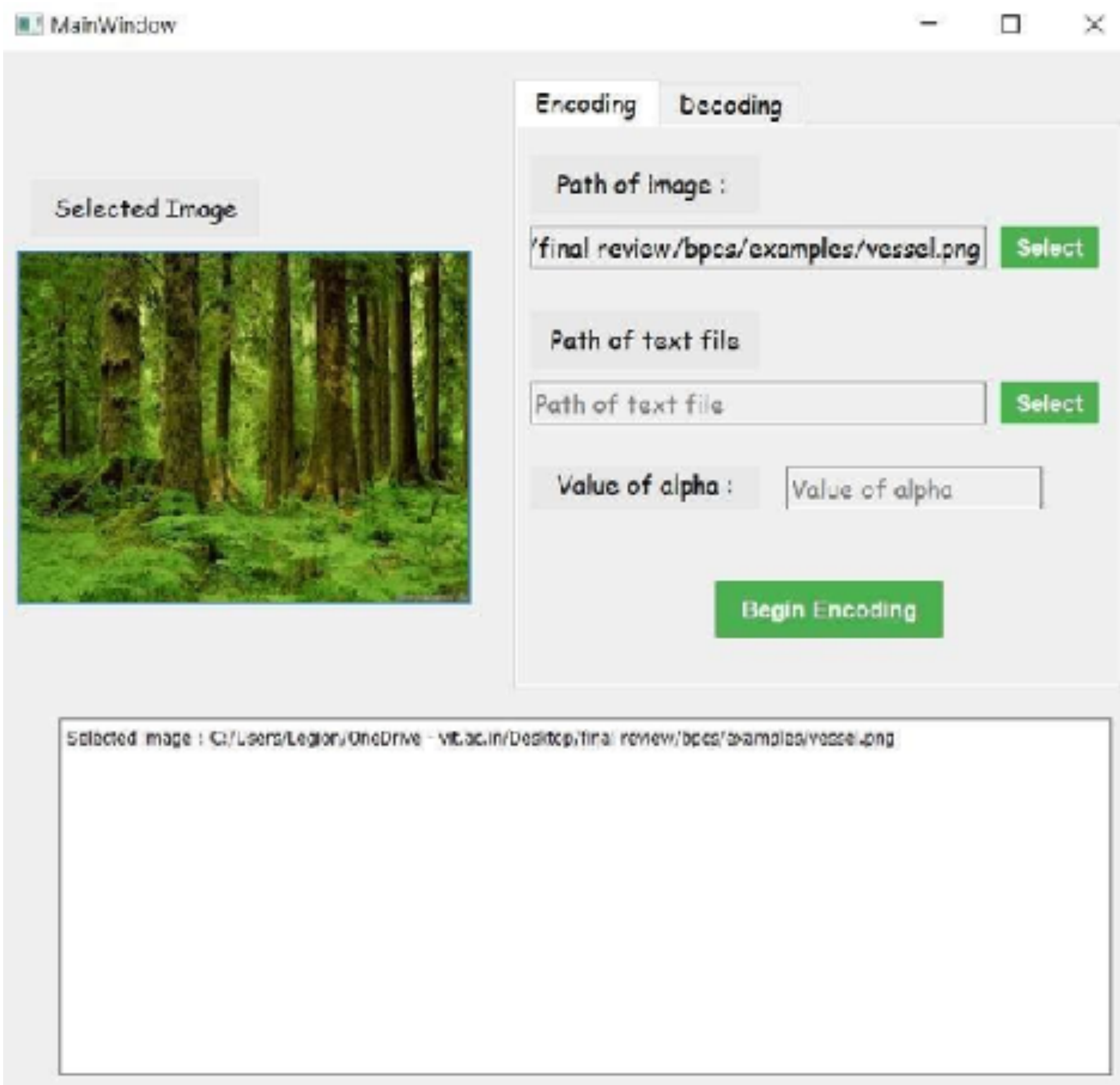
```
$ python -m bpcs.bpcs capacity -i
examples/vessel.png -a 0.45
```

Command Line Utility for Checking the Vessel Size

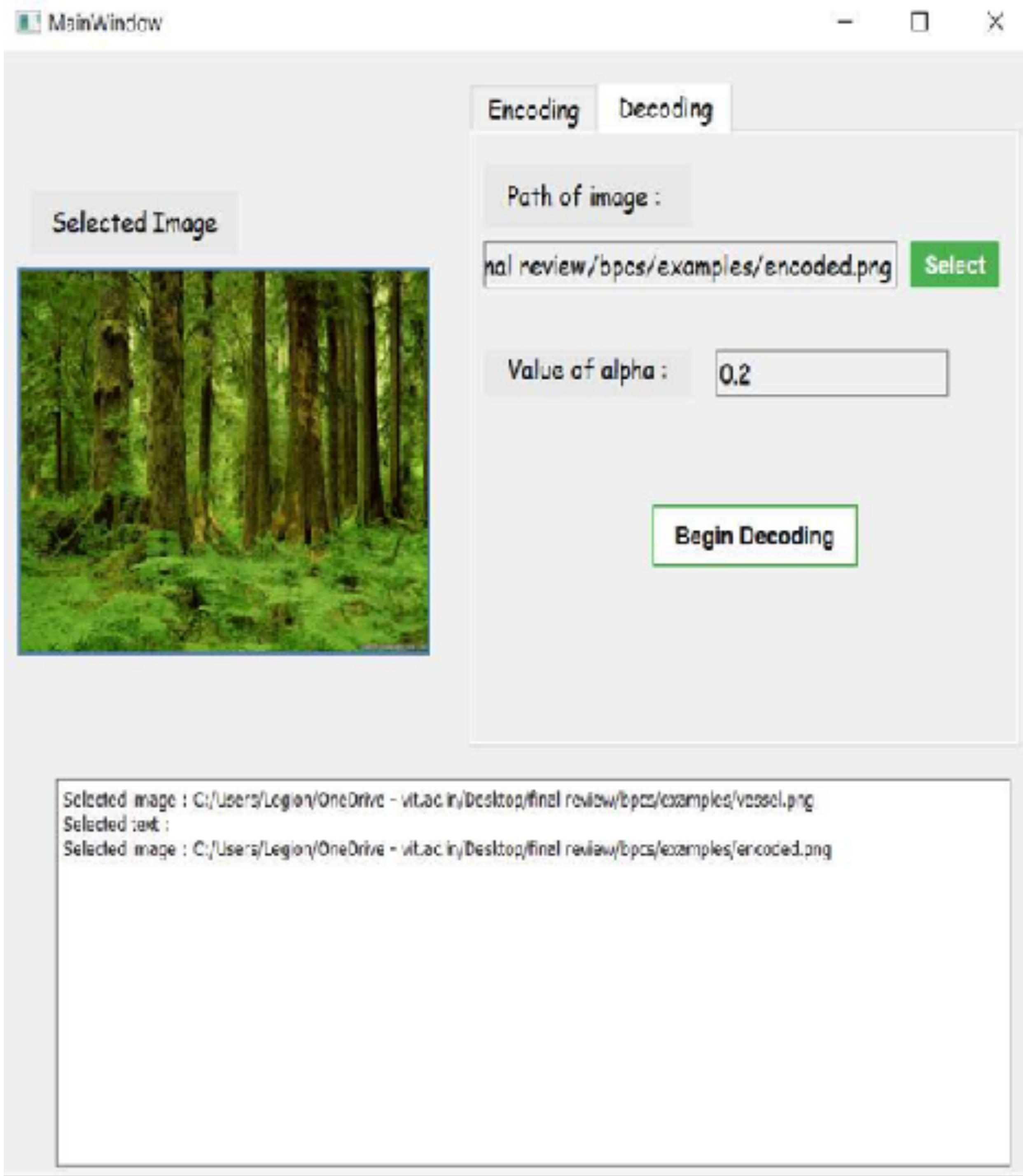
```
$ python -m bpcs.bpcs decode -i
examples/encoded.png -a 0.45 -o
examples/message_decoded.txt
```

Command line Utility for Decoding

GUI FOR DECODING and Encoding

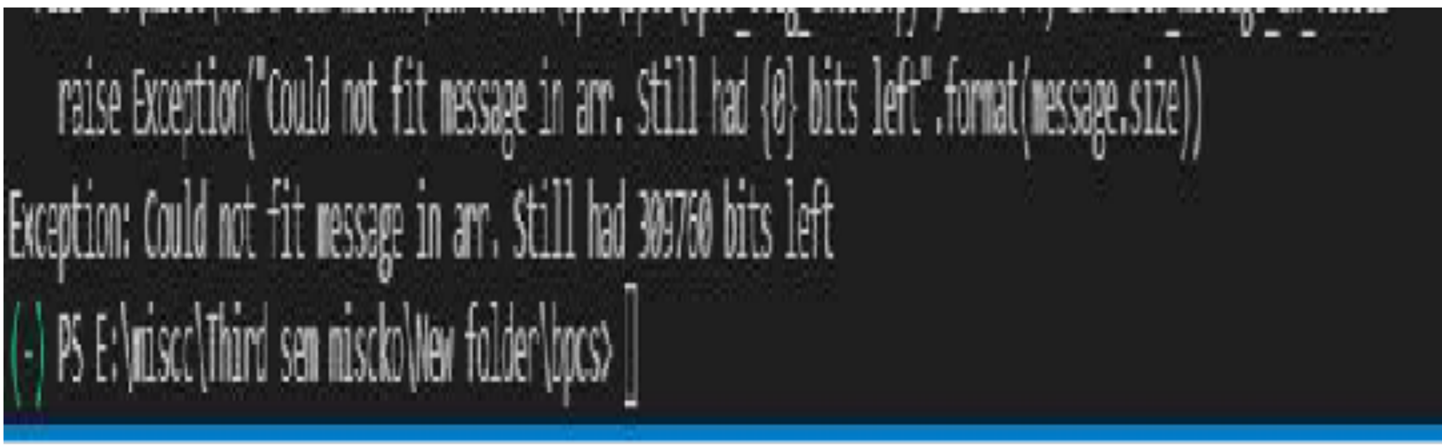


GUI for Encoding



<u>GUI for decoding</u>

We can view the size of text that can be encoded with the help of alpha value and our project throws exception if the text cannot be stored



ENCODING CODE SECTION

```
import numpy as np

from .logger import log
from .act_on_image import ActOnImage
from .array_message import read_message_grids, get_next_message_grid_sized
from .array_grid import get_next_grid_dims
from .bpcs_steg import arr_bpcs_complexity, conjugate

ALIVE, CONJUGATING, DEAD = 0,1,2
def get_message_and_status(message, dims, conjugated, status, alpha):
    """
    message is remaining message to be embedded
    dims is shape of next desired grid
    conjugated is list of bool, specifying whether each past message grid was conjugated or not
    status is int
        DEAD means we shouldn't be embedding any more messages
        ALIVE means we need the next grid from message
        CONJUGATING means we need the next grid from conjugated
    """
    if status == DEAD:
        return None, None, None, DEAD
    elif status == CONJUGATING and len(conjugated) == 0:
        return None, None, None, DEAD
    elif (status == ALIVE and message.size == 0) or (status == CONJUGATING):
        cur, conjugated = get_next_message_grid_sized(np.array(conjugated), dims, min_alpha=alpha)
        conjugated = conjugated.tolist()
        return cur, None, conjugated, CONJUGATING
    elif status == ALIVE:
        cur, message = get_next_message_grid_sized(message, dims)
        return cur, message, conjugated, ALIVE
```

```python
conjugated = []
status = ALIVE
nmessgs, nmaps, nleft, ngrids = 0, 0, 0, 0
for dims in get_next_grid_dims(arr, grid_size):
    ngrids += 1
    grid = arr[tuple(dims)]
    if arr_bpcs_complexity(grid) < alpha:
        nleft += 1
        continue
    cur_message, message, conjugated, status = get_message_and_status(message, grid.shape, conjugated, status, alpha)
    if status == DEAD:
        # since there is no more embedding to do, flip the remaining grids you find
        # so that they are not mistaken for signal grids when decoding
        cur_message = np.zeros(grid.shape, dtype=np.uint8)
        if not arr_bpcs_complexity(cur_message) < alpha:
            a = arr_bpcs_complexity(grid)
            b = arr_bpcs_complexity(cur_message)
            raise Exception('Error fixing vessel grid to have complexity below alpha: {0} -> {1}'.format(a, b))
        nleft += 1
    if status == ALIVE and arr_bpcs_complexity(cur_message) < alpha:
        cur_message = conjugate(cur_message)
        if not arr_bpcs_complexity(cur_message) >= alpha:
            a = arr_bpcs_complexity(conjugate(cur_message))
            b = arr_bpcs_complexity(cur_message)
            raise Exception('Error fixing message grid to have complexity above alpha: {0} -> {1}'.format(a, b))
        nmessgs += 1
        conjugated.append(True)
    elif status == ALIVE:
        nmessgs += 1
        conjugated.append(False)
```

```python
            nmaps += 1
        assert cur_message.shape == grid.shape
        arr[tuple(dims)] = cur_message
    if message is not None and message.size > 0:
        raise Exception("Could not fit message in arr. Still had {0} bits left".format(message.size))
    elif status != DEAD:
        raise Exception("Could not fully embed conjugation head in arr.")
    nfound = nmessgs + nmaps + nleft
    assert nmessgs + nmaps + nleft == ngrids, '{0} + {1} + {2} = {4} != {3}'.format(nmessgs, nmaps, nleft, ngrids, nfound)
    log.critical('Embedded {0} message grids and {1} conjugation maps'.format(nmessgs, nmaps))
    return arr

class BPCSEncodeImage(ActOnImage):
    def modify(self, messagefile, alpha):
        new_arr = np.array(self.arr, copy=True)
        message_grids = read_message_grids(messagefile, (8,8))
        return embed_message_in_vessel(new_arr, alpha, message_grids, (8,8))

def encode(infile, messagefile, outfile, alpha=0.45):
    x = BPCSEncodeImage(infile, as_rgb=True, bitplane=True, gray=True, nbits_per_layer=8)
    arr = x.modify(messagefile, alpha)
    x.write(outfile, arr)
```

DECODING CODE SECTION

```python
import numpy as np

from .logger import log
from .array_grid import get_next_grid_dims
from .act_on_image import ActOnImage
from .array_message import write_conjugated_message_grids
from .bpcs_steg import arr_bpcs_complexity

def remove_message_from_vessel(arr, alpha, grid_size):
    messages = []
    nfound, nkept, nleft = 0, 0, 0
    complexities = []
    for dims in get_next_grid_dims(arr, grid_size):
        nfound += 1
        grid = arr[tuple(dims)]
        cmplx = arr_bpcs_complexity(grid)
        if cmplx < alpha:
            nleft += 1
            continue
        complexities.append(cmplx)
        nkept += 1
        messages.append(grid)
    assert nfound == nkept + nleft
    log.critical('Found {0} out of {1} grids with complexity above {2}'.format(nkept, nfound, alpha))
    return messages

class BPCSDecodeImage(ActOnImage):
    def modify(self, alpha):
        return remove_message_from_vessel(self.arr, alpha, (8,8))
```

```python
            messages.append(grid)
    assert nfound == nkept + nleft
    log.critical('Found {0} out of {1} grids with
    return messages

class BPCSDecodeImage(ActOnImage):
    def modify(self, alpha):
        return remove_message_from_vessel(self.ar

def decode(infile, outfile, alpha=0.45):
    x = BPCSDecodeImage(infile, as_rgb=True, bitp
    grids = x.modify(alpha)
    write_conjugated_message_grids(outfile, grids
```

IMAGE PLOTTING CODE SECTION

```
import os
import subprocess
import numpy as np
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.pyplot as plt

from .text_to_image import str_to_words, get_word_color_map_fcn

def randrange(vmin, vmax):
    return (vmax-vmin)*np.random.rand(1) + vmin

def get_word_colors(infile):
    txt = open(infile).read()
    words = str_to_words(txt, True)
    get_color = get_word_color_map_fcn(words)
    for word in words:
        yield word, get_color(word)
    # return dict((word, get_color(word)) for word in words)

def hist(infile):
    words = open(infile).read().split()
    base = sorted(list(set(words)))
    inds = [base.index(word) for word in words]
```

```python
def plot(infile):
    fig = plt.figure()
    ax = fig.add_subplot(111, projection='3d')
    ax.set_xlabel('R')
    ax.set_ylabel('G')
    ax.set_zlabel('B')

    n = 100
    c, m = ('r', 'o')
    for word, (r,g,b) in get_word_colors(infile):
        ax.scatter(r, g, b, c=c, marker=m)
        if word.strip():
            print word.strip()
            subprocess.os.popen('say ' + word.strip())
        plt.pause(0.01)
        plt.draw()

def main(infile):
    hist(infile)
    # plot(infile)

if __name__ == '__main__':
    infile = 'docs/tmp.txt'
    main(infile)
```

FEW CODE SECTIONS FOR UI

```
<?xml version="1.0" encoding="UTF-8"?>
<ui version="4.0">
 <class>MainWindow</class>
 <widget class="QMainWindow" name="MainWindow">
  <property name="geometry">
   <rect>
    <x>0</x>
    <y>0</y>
    <width>799</width>
    <height>742</height>
   </rect>
  </property>
  <property name="windowTitle">
   <string>MainWindow</string>
  </property>
  <property name="styleSheet">
   <string notr="true">background-color: rgb(239, 239, 239);</string>
  </property>
  <widget class="QWidget" name="centralwidget">
   <widget class="QTabWidget" name="tab_widget">
    <property name="geometry">
     <rect>
      <x>360</x>
      <y>20</y>
      <width>430</width>
      <height>430</height>
     </rect>
    </property>
    <property name="font">
     <font>
```

```
    <x>360</x>
    <y>20</y>
    <width>430</width>
    <height>430</height>
   </rect>
  </property>
  <property name="font">
   <font>
    <family>Comic Sans MS</family>
    <pointsize>10</pointsize>
   </font>
  </property>
  <property name="currentIndex">
   <number>1</number>
  </property>
  <widget class="QWidget" name="encoding_tab">
   <property name="font">
    <font>
     <family>Comic Sans MS</family>
     <pointsize>10</pointsize>
    </font>
   </property>
   <attribute name="title">
    <string>Encoding</string>
   </attribute>
   <widget class="QLineEdit" name="path_of_image">
    <property name="geometry">
     <rect>
```

```xml
    <string/>
   </property>
   <property name="dragEnabled">
    <bool>false</bool>
   </property>
   <property name="placeholderText">
    <string>Path of image</string>
   </property>
  </widget>
  <widget class="QLineEdit" name="path_of_text">
   <property name="geometry">
    <rect>
     <x>10</x>
     <y>180</y>
     <width>320</width>
     <height>30</height>
    </rect>
   </property>
   <property name="font">
    <font>
     <family>Comic Sans MS</family>
     <pointsize>10</pointsize>
     <italic>false</italic>
    </font>
   </property>
   <property name="autoFillBackground">
    <bool>false</bool>
   </property>
   <property name="styleSheet">
    <string notr="true"/>
```

```
            <family>HelveticaNeue-Light</family>
            <pointsize>10</pointsize>
            <weight>75</weight>
            <bold>true</bold>
            <underline>false</underline>
            <strikeout>false</strikeout>
           </font>
          </property>
          <property name="autoFillBackground">
           <bool>false</bool>
          </property>
          <property name="styleSheet">
           <string notr="true">QPushButton {
    font-family: &quot;HelveticaNeue-Light&quot;,
color : white;
background-color: #4CAF50;
border-radius: 2px;
border: 2px solid #4CAF50;
}

QPushButton:hover{
      background-color: white;
      color: black;
}
</string>
          </property>
          <property name="text">
           <string>Begin Encoding</string>
          </property>
          <property name="flat">
           <bool>false</bool>
```

Implementation of RSA for encryption

Function to Encrypt and Decrypt the image

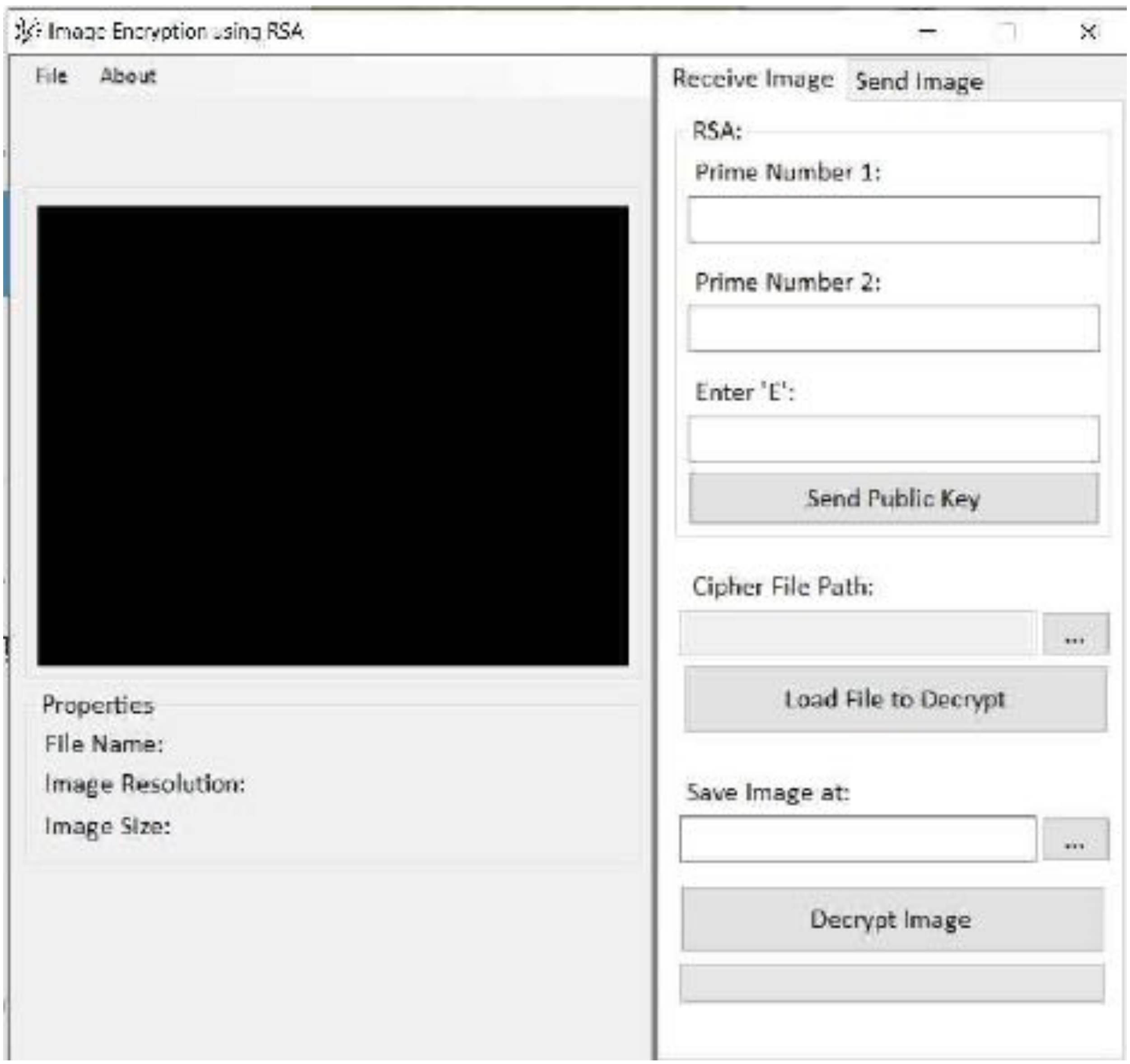
```
// Function to encrypt the image
private string EncryptImage(string imageHexToEncrypt)
{
  MessageBox.Show("RSA_E = " + _rsaE + "\nn = " + _n);
  var imageHex = imageHexToEncrypt;
  var imageHexArray = imageHex.ToCharArray();
  var cond = "";
  Progressbar_encryptImage.Maximum = imageHexArray.Length;
  for (var i = 0; i < imageHexArray.Length; i++)
  {
    Application.DoEvents();
    Progressbar_encryptImage.Value = i;
    if (cond == "")
      cond = cond + RSAalgorithm.BigMod(imageHexArray[i], _rsaE, _n);
    else
      cond = cond + " " + RSAalgorithm.BigMod(imageHexArray[i], _rsaE, _n);
  }

  return cond;
}
```

```
// Function to decrypt the image
private string DecryptImage(string imageToDecryptHex)
{
    var ImageHex = imageToDecryptHex.ToCharArray();
    var i = 0;
    var decryptResponse = "";
    ProgressBar_decrypt.Maximum = ImageHex.Length;
    try
    {
        for (; i < ImageHex.Length; i++)
        {
            Application.DoEvents();
            var c = "";
            ProgressBar_decrypt.Value = i;
            int temp;
            for (temp = i; ImageHex[temp] != '-'; temp++) c = c + ImageHex[temp];
            i = temp;
            var xx = Convert.ToInt32(c);
            decryptResponse = decryptResponse + (char)RSAalgorithm.BigMod(xx, _d, _n);
        }
    }
    catch (Exception ex)
    {
        Console.WriteLine(ex.Message);
    }

    return decryptResponse;
}
```

Code for Generation of Private key

```
namespace ImageCrypto
{
    internal static class RSAalgorithm
    {
        private static long Square(long a)
        {
            return a * a;
        }

        public static long BigMod(int b, int p, int m) //b^p%m=?
        {
            if (p == 0)
                return 1;
            if (p % 2 == 0)
                return Square(BigMod(b, p / 2, m)) % m;
            return b % m * BigMod(b, p - 1, m) % m;
        }

        public static int n_value(int prime1, int prime2)
        {
            return prime1 * prime2;
        }

        public static int cal_phi(int prime1, int prime2)
        {
            return (prime1 - 1) * (prime2 - 1);
        }

        public static int cal_privateKey(int phi, int e, int n)
        {
            int d;

            for (d = 1; ; d++)
            {
                var res = d * e % phi;
                if (res == 1) break;
            }

            return d;
        }
    }
}
```

Output Snapshots:



Here, after adding the suitable Prime numbers, public key is sent,

And private key is calculated on the sender side

Sender, gets the public key and loads the Image to Encrypt

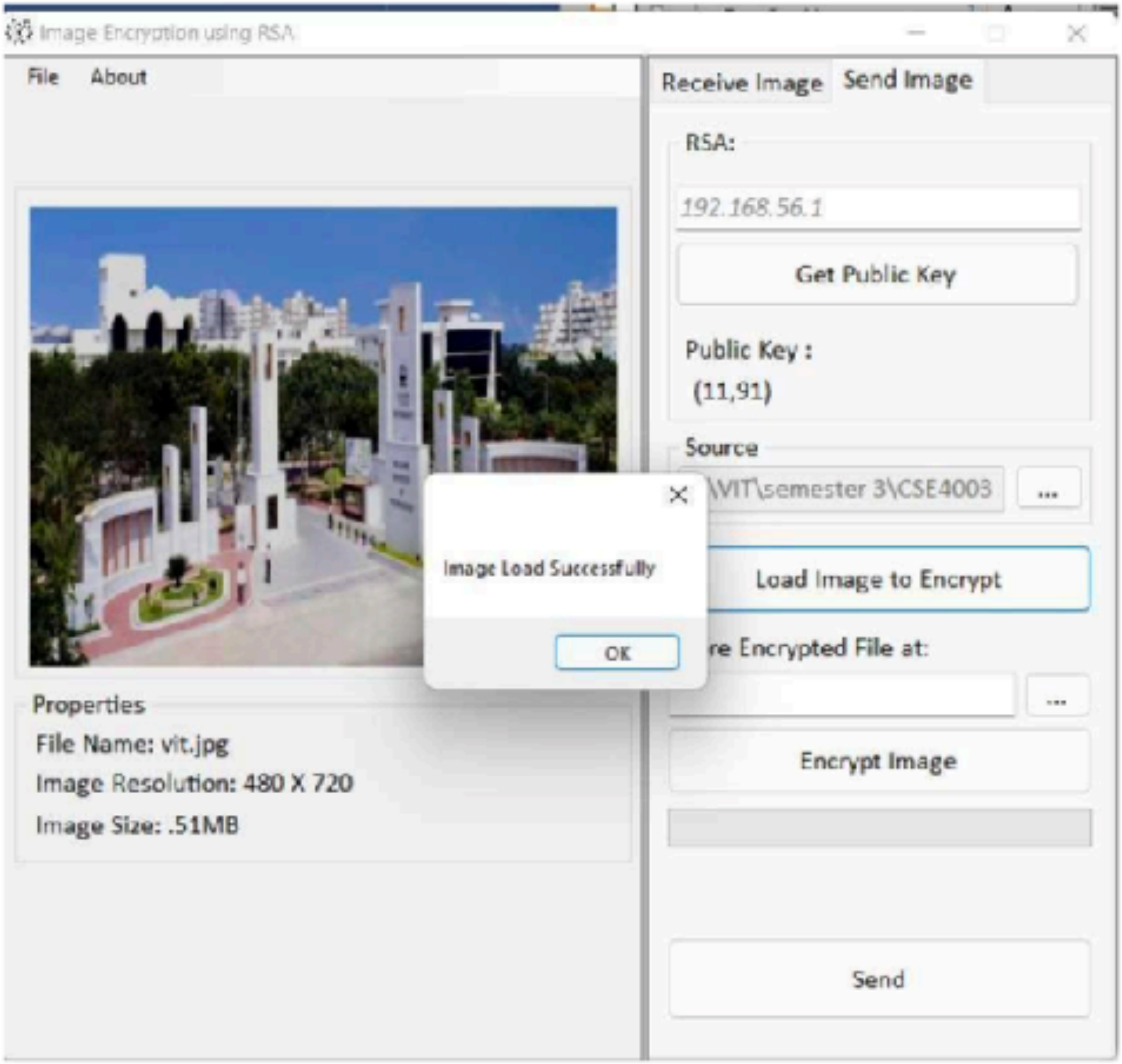


Image Encryption using RSA
File
About
Receive Image
Send Image
RSA:
192.168.56.1
Get Public Key
Public Key :
(11,91)
Source
\VIT\semester 3\CSE4003
...
Image Load Successfully
OK
Load Image to Encrypt
re Encrypted File at:
...
Encrypt Image
Send
Properties
File Name: vit.jpg
Image Resolution: 480 X 720
Image Size: .51MB

RESULTS

Thus, from the experiment for three different sets of images, it is concluded that the BPCS technique has high data embedding capacity in the range 50 – 60 %. Also, it is seen that the original image and the final embedded image appear to be identical to the human eye. This experiment has been carried on bitmap images. We can now experiment on other types of images like jpeg, tiff etc. and check the results. In this experiment, we used complexity technique based on length of black & white border. In future work, we can experiment using different complexity techniques and compare them based on the results obtained. Steganography and RSA could prove to be a very secured medium. BPCS with 50 percent embedding capacity and RSA, asymmetric public key crypto system both together can enhance security to great extent

CONCLUSION

Based on study of image encryption techniques we found that using bit rotation, reversal, mathematical models and matrix manipulation techniques we can encrypt images. These techniques can be very useful in medical imaging, space applications, and social media application. In existing methods, we found there can be vulnerable attacks on password secret key that we are using for bit rotation and reversal, extended hill cipher. In order to improve security of image data encryption we proposed algorithm that uses password generated using RSA algorithm. So due to this modification security of password key will be increased to brute force attack. This project introduces a novel steganographic approach for covert communications between two private parties. The approach introduced in this project makes use of both steganographic as well as cryptographic techniques. The process involves converting a Secret image into a text document, then encrypting the generated text into a cipher text using a key (Password) based encryption algorithm, and finally embedding the cipher text on to a cover image. This embedding process is carried out using a threshold-based scheme that inserts secret message bits into the cover image only in selected pixels.

FUTURE WORK/SCOPE :

The future scope of BPCS algorithm-based steganography with RSA integration involves several potential advancements:

- Increased Security: RSA encryption adds an additional layer of security to steganography by protecting the hidden information with strong encryption. Future developments may focus on enhancing the encryption algorithms, key generation techniques, and key management practices to ensure robust security.
- Improved Capacity and Efficiency: Research efforts may aim to increase the capacity of the stego-images while maintaining their visual quality. This can involve optimizing the embedding process, refining the bit-plane selection algorithms, or exploring alternative image formats that offer higher data-hiding capabilities.
- Adaptive and Intelligent Systems: Future systems may incorporate adaptive and intelligent techniques to dynamically adjust the embedding process based on image characteristics, content complexity, and potential detection vulnerabilities. Machine learning and artificial intelligence algorithms can be employed to enhance the system's robustness and effectiveness.
- Multi-Media Steganography: While BPCS steganography primarily focuses on image hiding, future scope may involve expanding the techniques to other multimedia formats such as audio and video. This can open up new possibilities for hiding sensitive information within different types of digital media.
- Resistance to Advanced Attacks: Researchers will likely explore novel attacks and vulnerabilities against BPCS steganography with RSA integration to develop countermeasures. This can involve analyzing potential weaknesses, improving the detection of stego-images, and devising methods to withstand advanced statistical analysis and machine learning-based attacks.
- Integration with Emerging Technologies: BPCS steganography with RSA integration can be combined with emerging technologies such as blockchain and distributed ledger systems to provide tamper-proof and decentralized steganographic solutions. This can enable secure communication and data hiding in various domains, including finance, healthcare, and IoT (Internet of Things).

Overall, the future scope of BPCS algorithm-based steganography with RSA integration lies in advancing security, capacity, efficiency, adaptability, and integration with emerging technologies. These advancements will contribute to the development of more robust and secure methods for concealing and transmitting sensitive information.

References:

1. Chawla, R. K. (2017). Watermarking Using Bit Plane Complexity Segmentation and Artificial Neural Network. *International Journal of Scientific Research and Management*. Published. https://doi.org/10.18535/ijsrm/v5i6.04
2. Hosam, O., & Ahmad, M. H. (2019). Hybrid design for cloud data security using combination of AES, ECC and LSB steganography. *International Journal of Computational Science and Engineering*, *19*(2), 153. https://doi.org/10.1504/ijcse.2019.100236
3. Ji, S. S. (2015). A Study of Optimal Image Steganography based on LSB Techniques. *Journal of the Korea Industrial Information System Society*, *20*(3), 29–36. https://doi.org/10.9723/jksiis.2015.20.3.029
4. Kaushal, S., & Kakkar, A. (2018). LSB based Steganography Techniques for Secured Communication. *International Journal of Computer Applications*, *181*(10), 28–31. https://doi.org/10.5120/ijca2018917651
5. A literature Review of Cryptography on RC4 Stream Cipher and Hash-LSB Steganography. (2020). *International Journal for Research in Engineering Application & Management*, 329–332. https://doi.org/10.35291/2454-9150.2020.0308
6. Pavan Kumar, A., Gajjela, L., & Sai, N. R. (2020). A Hybrid Hash-Stego for Secured Message Transmission Using Steganography. *IOP Conference Series: Materials Science and Engineering*, *981*(2), 022014. https://doi.org/10.1088/1757-899x/981/2/022014

7. Sharma, H. (2013). Secure Image Hiding Algorithm using Cryptography and Steganography. *IOSR Journal of Computer Engineering*, *13*(5), 01–06. https://doi.org/10.9790/0661-1350106
8. Singla, G., & Singh, C. (2019). A Review on Steganography Data Hiding using Color Images. *International Journal of Trend in Scientific Research and Development*, *Volume-3*(Issue-4), 889–893. https://doi.org/10.31142/ijtsrd23556