Visual Cryptography in Proof of Ownership

Abstract

In the modern world, claiming true ownership of images that you take has become increasingly difficult with the advent of modern image manipulation methods. Even when one contests ownership - it goes on to be very hard to prove it in a court of law or any other platform since the similarities between images require visual comparison which comes with its own ambiguity and vague nature. The aim of this project is to discuss and implement a system which can be used to prove ownership over an image using visual cryptographic techniques

Introduction

Today, millions of images are distributed on the internet. As a result, it has become important to claim ownership of the very images that you take. In this project, we aim to achieve that using two core concepts - Visual Cryptography and Sampling distribution of means. Visual Cryptography refers to the process of hiding any number of images using shadow images which are referred to as shares. Sampling distribution is a simple mathematical method which will aid us in achieving our end goal by exploiting the way images work in digital media, primarily in its pixel format.

The system aims to use a signing image or a secret image as leverage to prove a real connection between any image taken and the entity that was involved in the original creation of the image. This is accomplished with the use of a text based secret seed key as well as sampling distributed means from the image that is to be protected and using the combination of the two to obtain multiple shares that can be used to prove ownership over the image. These shares can then be stored with third parties and can be invoked at a later date to prove ownership over the images.

Related work or Literature Survey

S.NO	PAPER TITLE	METHODS	DRAWBACKS	ADVANTAGES
1.	Survey of Visual Cryptography Schemes.	This paper discusses several visual cryptography schemes and evaluates performance based on several criteria like pixel expansion, image format etc.	No noticeable drawbacks	1. Provides an insight into black and white visual cryptography schemes like sharing single and multiple secrets. 2. Discusses several visual cryptography schemes that involve sharing colored secret images with arcs.
2.	An Authentication System For Online Question Paper Delivery using Visual Cryptography	This paper discusses an authentication system that eliminates drawbacks like malpractice and congestion in mobile networks.	Only a single secret image can be shared with the traditional VC technique. Some techniques involve shares to be properly aligned.	1. With VCTSTS algorithm, two password images can be encrypted with only two shares. 2. Reduces malpractice and congestion in mobile networks because shares are sent in the form of email attachment.

3.	Probabilistic Grayscale Visual Cryptography Scheme Using Multi-Pixel Encoding	This paper discusses a multi-pixel encoding method based on GPM and schemes that help to improve the perceptual quality of images in GPVCS.	No noticeable drawbacks	1. Discusses two ways to obtain a GPM, top-down approach (which involves solving multi objective model) and bottom-up model (that constructs GBM or GPM)
4.	Enhancing Security of Image Steganography Using Visual Cryptography	This paper discusses the LSB method and RGB to perform the steganography operation. It also proposes a different method rather than the traditional use of passwords and cryptography for security.	No noticeable drawbacks	1. Uses visual cryptography to propose a new method for carrying out steganography operation. 2. This method uses image and text as a secret message, and improves the security of the system.
5.	Visual cryptography techniques: Short Survey	This paper discusses different methods for visual cryptography and discusses five criteria: numbers of input pictures, pixel expansion, Formats of image, type of shares	Some drawbacks in the colour image scheme involve great pixel expansion, reflecting a small number of colors.	1. Discusses various methods and applications of visual cryptography 2.Discusses various techniques based on five criteria like, numbers of input pictures, pixel expansion,

		produced, and quality of a reconstructed image		Formats of image, type of shares produced, and quality of a reconstructed image.
6.	Image Security using Visual Cryptography	A public key encryption method is discussed that helps to make the picture exchanges in visual cryptography more secure	No noticeable drawbacks	1.This paper discusses that the ability to decipher the secret image by superimposing shares without calculation is a distinguishing feature of the Visual Cryptography Scheme.
7.	An Implementation of Algorithms in Visual Cryptography in Images	NA	No noticeable drawbacks	Discusses splitting of the image into shares that are then given to participants, the image is obtained by stacking the shared images.

Proposed System

Overview

People are regularly sharing millions of images on the internet. As a result, it has become important to claim ownership. There are two concepts involved that help to achieve this, Visual Cryptography and Sampling distribution of means. Visual Cryptography refers to the process of hiding any number of images using shadow images which are referred to as shares.
The system primarily makes use of two images which will be henceforth referred to as main and secret images. The main image is the image that you wish to have proof of ownership of - it could be a picture you took or a piece of art that you create. The secret image would be the aforementioned 1 bit depth black and white logo image. The core idea behind the system is to create multiple shares (2 in our case) of the secret image which can be used to prove ownership later on. This is done by using an OTP algorithm along with sampling of the main image done using a pseudorandom technique which would depend on a user given key. This key is also important as without it - the process cannot be recreated in the same fashion and therefore this key also happens to behave like a symmetric key.
The process begins by creating a master share by first converting the main image to black and white (this ensures consistent processing and true originality testing even after malicious parties employ color shifting methods). Once a BW image is obtained, we find the mean value of all of its pixels, going forward this will be referred to as MPV (Mean Pixel Value). Now we begin to loop over every single pixel of the main image and for each pixel - a set of N pixels would be picked from the main image itself based on a pseudorandom algorithm which depends on the user input key. The mean for these N pixels would be calculated and compared against the MPV and if the value is less then Fig 1 would be used in the output master share and if the value is more Fig 2 would be used. To clarify, each pixel would be replaced by 4 pixels belonging to one of the 2 patterns below. As evident, the output image would contain 4 times as many pixels as the input image - this makes the master share generation process highly CPU intensive and takes longer time as the pixel count of the image increases.

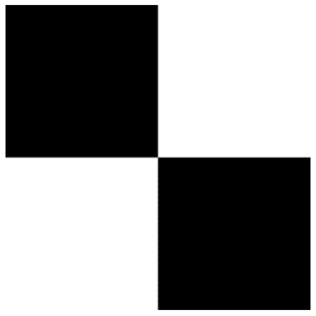


Fig 1

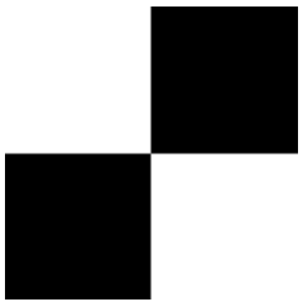
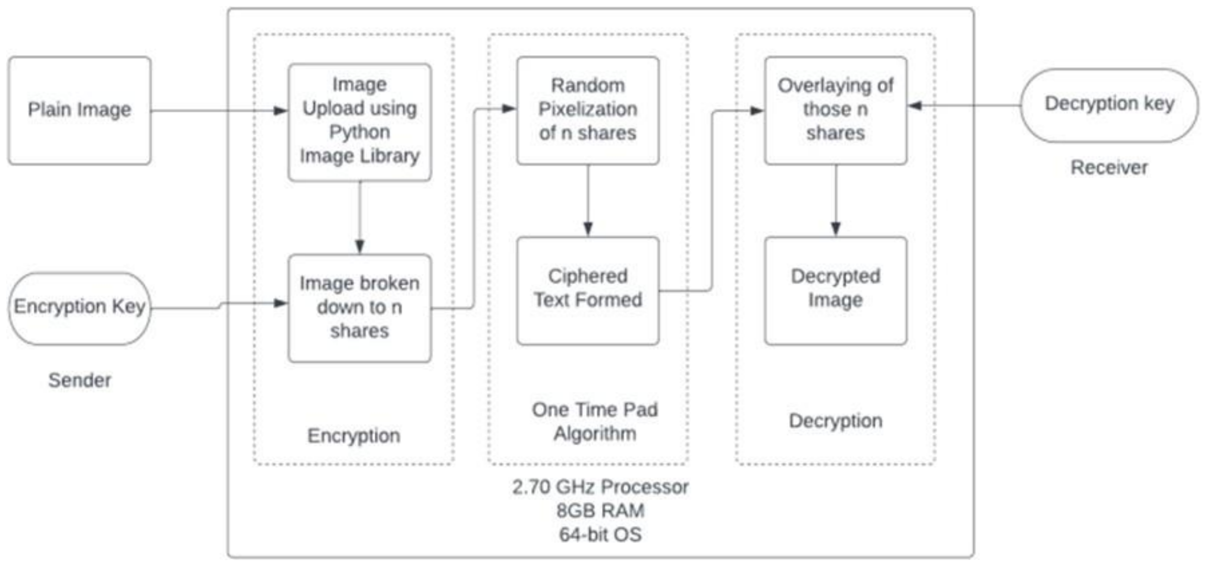


Fig 2

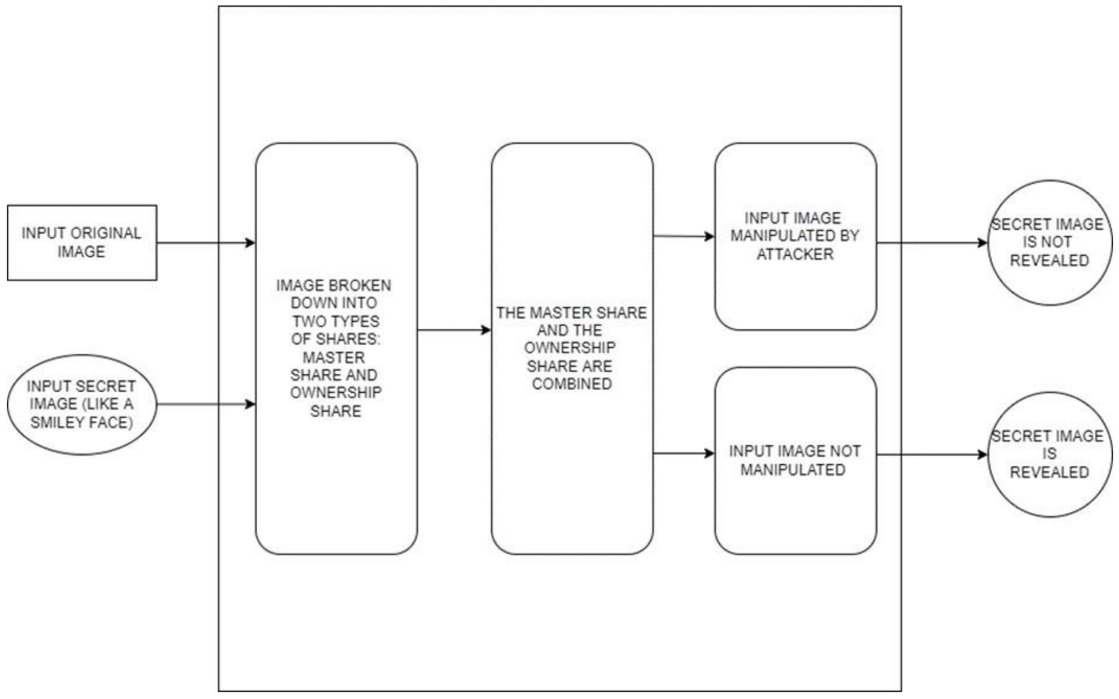
The second step of the process is generation of ownership share. In this step, the secret image is utilized to create the ownership share. To start, we loop through the pixels of the secret image and then for each pixel we sample pixels from the main image using the same pseudo random algorithm using the same user input key. We again compare Mean Pixel Values and determine which configuration of the 4 pixels to replace the pixel with. Once done, we would have the ownership share in our hands. The ownership share and master share can be laid on top of one another to show the secret image in real life. This allows for the owner of the photo to submit their ownership share to a central authority and have them keep track of its relationship with the image on their databases. When you believe another party is unrightfully using your images, you can claim ownership by pointing the central authority toward the unrighteous party's image and secretly provide the central authority with your signing key. Assuming the stolen image is actually yours, the central party can utilize your key to generate a new master share based off of the stolen image - and if the generated mastershare is overlaid on top of the ownership share you submitted long back results in your secret image showing up, that is enough proof for the central authority to back your claim of ownership.

This method even works if the stolen image is modified using blurs, angle shifting, color shifting and other common editing methodologies. If this method cannot prove ownership, then it is safe to say that the image is modified beyond recognition to the point where it is a completely different image and therefore losing all ties to the original image.

System Architecture



Functional Architecture


INPUT ORIGINAL IMAGE
INPUT SECRET IMAGE (LIKE A SMILEY FACE)
IMAGE BROKEN DOWN INTO TWO TYPES OF SHARES: MASTER SHARE AND OWNERSHIP SHARE
THE MASTER SHARE AND THE OWNERSHIP SHARE ARE COMBINED
INPUT IMAGE MANIPULATED BY ATTACKER
INPUT IMAGE NOT MANIPULATED
SECRET IMAGE IS NOT REVEALED
SECRET IMAGE IS REVEALED

Modular Design

1. *Load Module*

In this module, the loading of input images takes place.

2. *OTP Module*

This module involves generation of a secret image with a one-time validity

3. *Input Verification Module*

This module verifies the input image and makes sure that it has the valid dimensions

4. *Image Decryption Module*-

In this module, the deciphering of the image is carried out using the secret image

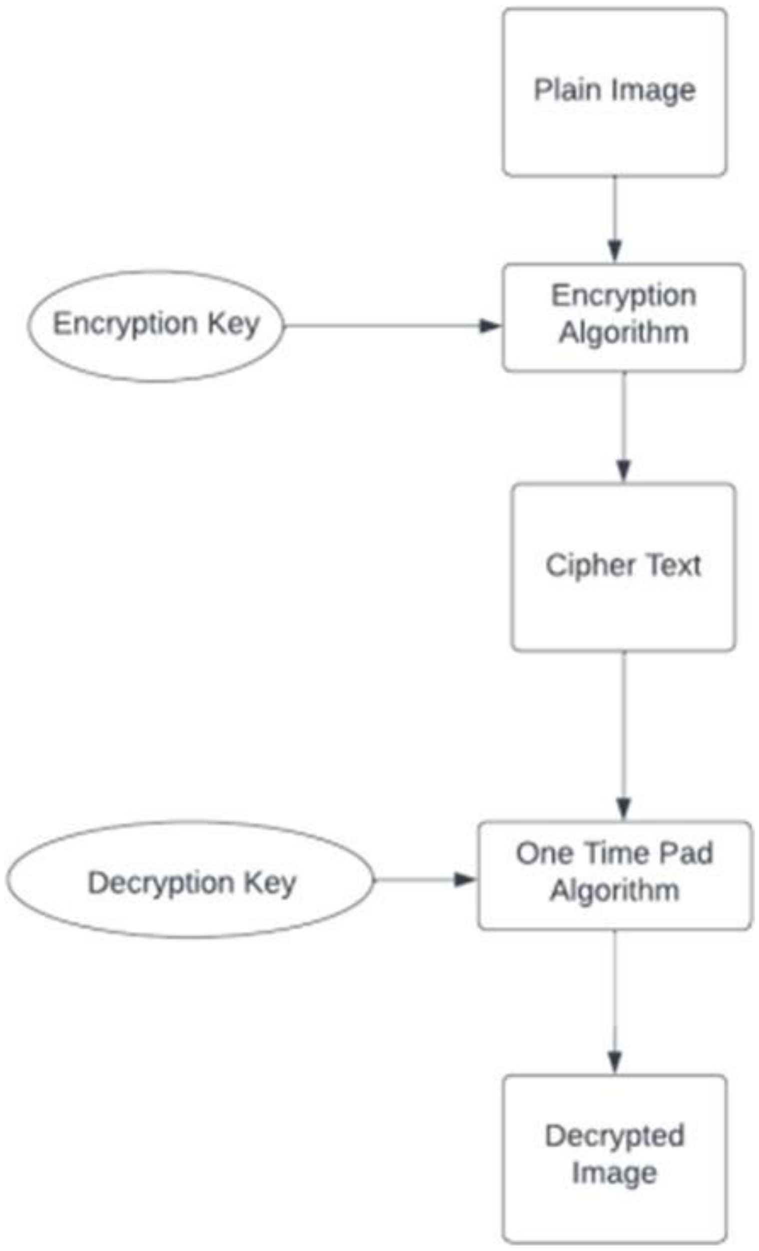
5. *Message Image Preparation*

This module involves resizing the image if it has incorrect dimensions.

6. *Image Ciphering Module* –

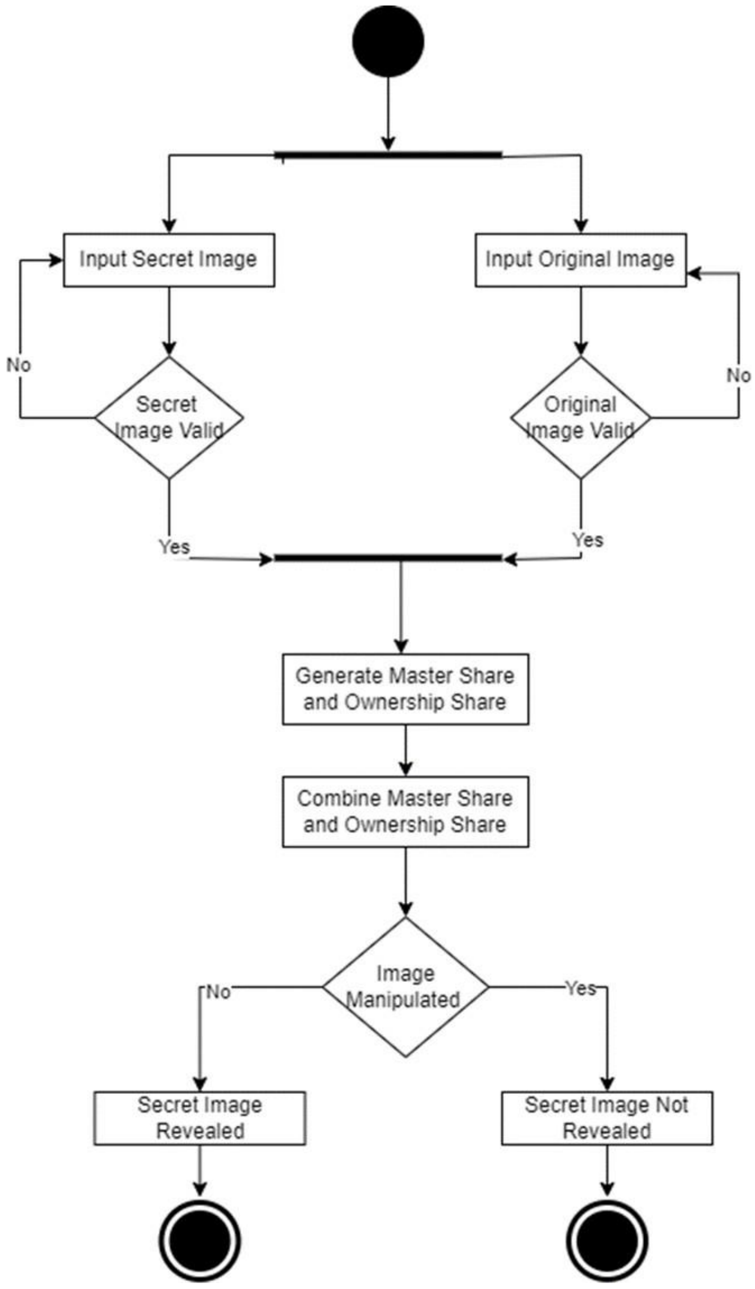
In this module, pixel shuffling is carried out to cipher the input image.

Flow Diagram



Activity Diagram

This figure shows two inputs, input secret image and input original image. It can be used to generate two types of shares namely, master share and ownership share. A secret image can be anything like a smiley face. Now, consider a situation where someone steals the original image and manipulates it. In that case on combining both the shares (i.e., the ownership and master share), the secret image won't be revealed.



Innovative Idea

In a modern world, where hackers and other malicious entities plague the open internet with various security threats and data breaches, being able to move sensitive information and proof of ownership documents to an analogue world and yet being able to automate these processes comes in as a great advantage. Using the proposed system, we can actually shift the storage of the proof of ownership shares by printing it on a transparent sheet and store it in the real world for safekeeping and simply print out master shares to overlay on top of them to prove ownership.
There are 3 general visual cryptography algorithms: Input Image Verification Algorithm, Image (Message) Encryptions Algorithm, and Ciphered and Key Image Alignment (used for decryption), that are useful in a variety of applications like Authentication, Digital Signatures, Secure Network Communications, Time Stamping, Disk Encryption etc. However, apart from these algorithms, a unique algorithm was in use too which is called the One-time pad algorithm. This method involves the encryption of a key with a one-time validity. This key either has the same size or larger size than the message to be encrypted and is known as One-time pad. The next step involves pairing of a random secret key with the message to be encrypted, followed by combining with modular addition to perform message encryption. This algorithm has various advantages like, it is easier to compute and carry out Bitwise XOR operation. Secondly, it is a very secure algorithm and regardless of the resources that the attacker possesses, the plaintexts look equally likely.

Implementation Details and Analysis

Software Details

The technical side of this project uses python as its main programming language. To handle image processing, we made use of Pillow or PLI (Python Image Library) which provided simple APIs to handle image manipulation. Numpy was essential to handle matrix processing for the images and last but not least, the default random library was utilized to accomplish the pseudo random pixel selection.

Looping through pixels was accomplished by converting images into 2 dimensional matrices and then flattened the 2D matrix into a 1D matrix and then iterating over that list. This meant that the algorithm effectively worked at an O(n) complexity where n is the total number of pixels present in the images. However, it is pertinent to point out that this process is parallelizable if approached correctly. Incorrectly parallelizing it might result in not being able to reproduce the same master share later on with a stolen image due to the inherent nature of the pseudo random selector. Speaking of, the way we managed to implement a pseudorandom selector is by exploiting the way the random function in python’s default library works. The algorithm it uses internally is inherently a pseudo random algorithm for selection based on a seed value. In regular runtime, the seed value is the clock time at which the program began executing. However, in our application, the seed is provided by the user as a symmetric secret key therefore allowing the system to be able to reproduce the pseudo random selection consistently every time as long as the same key is provided.

Sample Code

Split code:

```
from PIL import Image, ImageDraw
import os
import random
import argparse
from numpy import mean
"""
Terms used:
payload - Image that will be split into 2
key     - Image that will be used to aid in the process of splitting
seed    - A string value that will be used in the process of sample
selecting within splitting
"""
def get_options():
    parser = argparse.ArgumentParser()
    parser.add_argument('--payload', '-p', required=True)
    parser.add_argument('--key', '-k', required=True)
    parser.add_argument('--seed', '-s', required=True)
    return parser.parse_args()
def main():
    args = get_options()

    payload_file = args.payload
    key_file = args.key
    seed = args.seed
    # Same seed will result in same random behavior
    random.seed(seed)
    # Arbitrary - but must be consistent in production use
    # Should be less than total pixels in payload image
    SAMPLE_PIXEL_COUNT = 10
    # Checking if payload file exists
    if not os.path.isfile(payload_file):
        print("The splittable file does not exist.")
        exit()
    if not os.path.isfile(key_file):
        print("The key file does not exist.")
        exit()
    payload_image = Image.open(payload_file)
    key_image = Image.open(key_file)
    f, _ = os.path.splitext(payload_file)
    out_filename_A = "A.png"
    out_filename_B = "B.png"

    payload_image = payload_image.convert('1')  # convert image to 1 bit
    key_image_greyscale = key_image.convert('LA')
    greyscale_pixel_values = list(key_image_greyscale.getdata())
    # pixel_values is an array of tuples (2D), we flatten it to get an
array (1D)
    greyscale_pix_values_flat = [
        x for sets in greyscale_pixel_values for x in sets]
    mean_pixel_value_key_image = mean(greyscale_pix_values_flat)
    print("Image size: {}".format(payload_image.size))
    # Prepare two empty slider images for drawing
    width = payload_image.size[0]*2
    height = payload_image.size[1]*2
    print("Output size: {} x {}".format(width, height))
    out_image_A = Image.new('1', (width, height))
    out_image_B = Image.new('1', (width, height))
    draw_A = ImageDraw.Draw(out_image_A)
    draw_B = ImageDraw.Draw(out_image_B)
    # Two possible patterns to choose from
    patterns = ((1, 0, 1, 0), (0, 1, 0, 1))
    # Cycle through pixels
    for x in range(0, int(width/2)):
        for y in range(0, int(height/2)):
            # Pattern selection
            # This is done by selecting picking a sample of pixels from
payload image's greyscale version
            # Selection is done based on seed provided so same seed
results in same selections
            key_array_len = len(greyscale_pix_values_flat)
            pixels_to_sample = (SAMPLE_PIXEL_COUNT + x + y) %
key_array_len
            # Making sure the value isn't 0
            if (pixels_to_sample == 0):
                pixels_to_sample = SAMPLE_PIXEL_COUNT
            sampleMean = mean(random.sample(
                greyscale_pix_values_flat, pixels_to_sample))
            # Core decisional logic to choosing which pattern goes
            if sampleMean > mean_pixel_value_key_image:
                pat = patterns[0]
            else:
                pat = patterns[1]
            pixel = payload_image.getpixel((x, y))
            # A will always get the pattern
            draw_A.point((x*2, y*2), pat[0])
            draw_A.point((x*2+1, y*2), pat[1])
            draw_A.point((x*2, y*2+1), pat[2])
            draw_A.point((x*2+1, y*2+1), pat[3])
            if pixel == 0:  # Dark pixel so B gets the anti pattern
                draw_B.point((x*2, y*2), 1-pat[0])
                draw_B.point((x*2+1, y*2), 1-pat[1])
                draw_B.point((x*2, y*2+1), 1-pat[2])
                draw_B.point((x*2+1, y*2+1), 1-pat[3])
            else:
                draw_B.point((x*2, y*2), pat[0])
                draw_B.point((x*2+1, y*2), pat[1])
                draw_B.point((x*2, y*2+1), pat[2])
                draw_B.point((x*2+1, y*2+1), pat[3])
    out_image_A.save(out_filename_A, 'PNG')
    out_image_B.save(out_filename_B, 'PNG')
    print("Done.")
if __name__ == '__main__':
    main()
```

Overlay code:

```python
import os
from PIL import Image
import sys

if len(sys.argv) != 3:
    print("This takes two arguments; the two images to be combined.")
    exit()
fileA = str(sys.argv[1])
fileB = str(sys.argv[2])

if not os.path.isfile(fileA):
    print("The first file does not exist.")
    exit()

if not os.path.isfile(fileB):
    print("The second file does not exist.")
    exit()

infile1 = Image.open(os.path.join(
    fileA))
infile2 = Image.open(os.path.join(
    fileB))
outfile = Image.new('1', infile1.size)
for x in range(infile1.size[0]):
    for y in range(infile1.size[1]):
        outfile.putpixel((x, y), max(
            infile1.getpixel((x, y)), infile2.getpixel((x, y))))

outfile.show('decrypted.png')
```

Result

<u>Before Encryption</u>

Source Secret Image

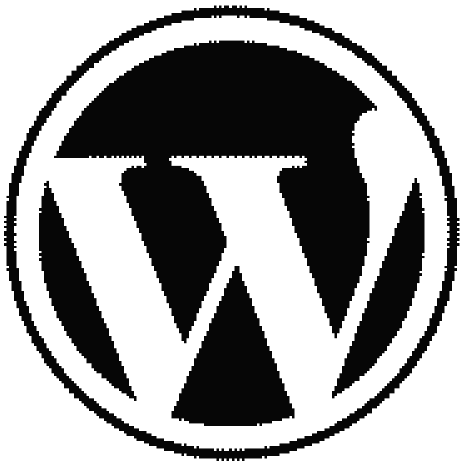


Fig 3

<u>Example of shares</u>

Master Share

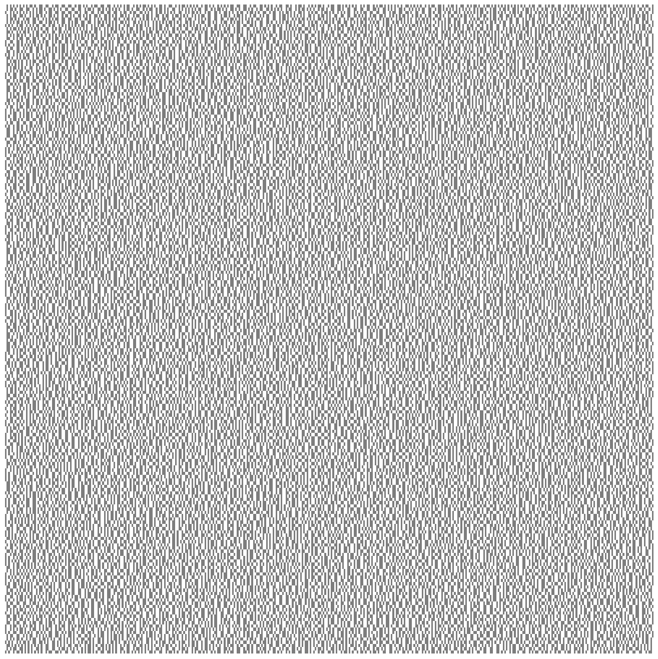


Fig 4

Ownership Share

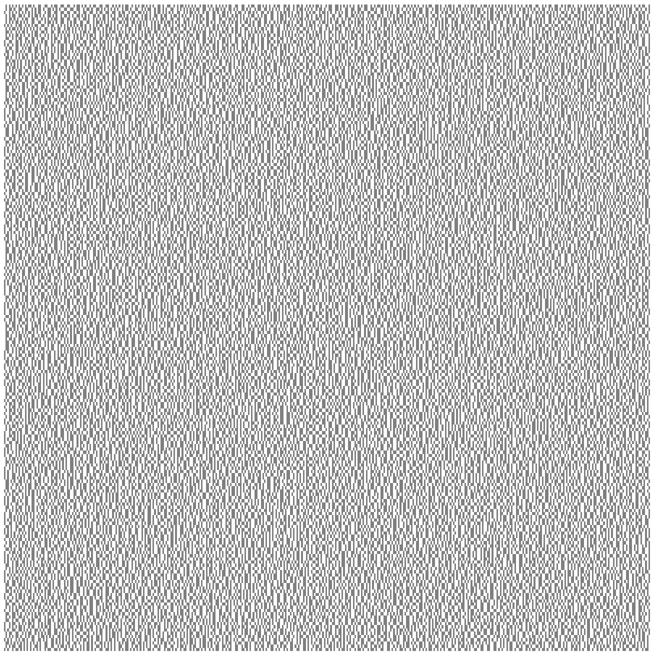


Fig 5

<u>After Encryption</u>

Shares when laid on top of each other (Digitally simulated)

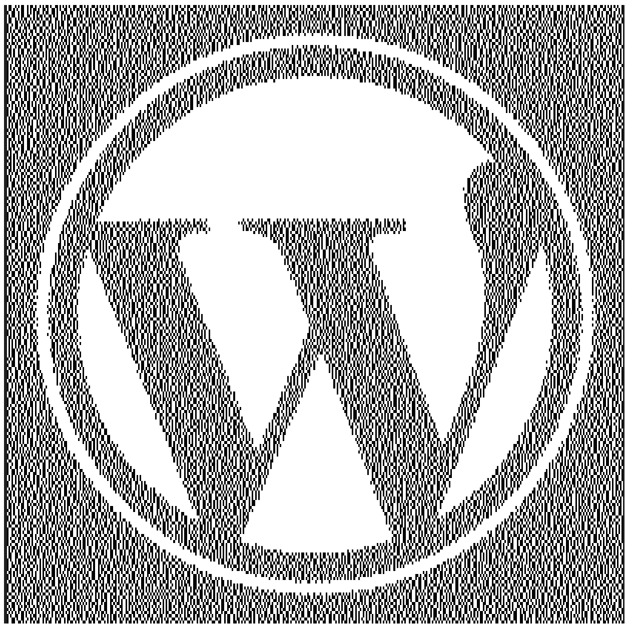


Fig 6

Conclusion and Future work

In this project, we discuss a unique way of encrypting an image by breaking it down into not one, but two noisy images, one which is generated at random and one which holds a cipher message that makes it very difficult for attackers to decipher. But, the existing system also has some disadvantages. The advantages are, it is easy to implement, low computation cost because the secret message is recognized only by human eyes and is not cryptographically computed; it is simple to incorporate; the desired receiver can see the image by combining the ciphered and one-time valid secret image. Moreover, this one-time validity helps make the cryptography safe from attackers.

The disadvantages are, it uses a single algorithm to encrypt the file and once the algorithm is cracked by attackers, they'll be able to access all the files that are yet to be shared, while if the random ciphered image is generated along with a secret image, knowing the algorithm itself is just not sufficient to decipher the yet to be shared files. The contrast of the reconstructed image is not maintained. Perfect alignment of the pixelations is difficult. Because of the pixel expansion the width of the decoded image is double the original image. Because of the change in aspect ratio, this results in a loss of data.

We have discussed a variety of ideas since the beginning. Visual Cryptography is being used by various countries to subtly move manually written archives, budgetary reports, content pictures, web casting a ballot and so forth. The current visual cryptography plans result in pixel extension and inadequate visual quality. However, further work to upgrade the visual cryptography component to ensure greater security of the data can be done. Coordinating VC plots with computerized watermarking, steganography could be improving the security level of the data. Also, in future, it can be used various in domains like Biometric security, Watermarking, Steganography, Remote electronic voting, Bank customer identification and many more.

References

1. Gyan Singh Yadav and Aparajita Ojha. ”A Novel Visual Cryptography Scheme Based on Substitution Cipher”. Proceedings of the 2013 IEEE Second International Conference on Image Information Processing (ICIIP-2013).

2. An Implementation of Algorithms in Visual Cryptography in Images, Archana B.Dhole and Prof. Nitin J. Janwe, 2013

3. Liu, F., Yan, W.Q. (2014). Various Problems in Visual Cryptography. In: Visual Cryptography for Image Processing and Security.

4. Kulvinder Kaur and Vineeta Khemchandani. ”Securing Visual Cryptographic Shares using Public Key Encryption”.2013 3rd IEEE International Advance Computing Conference (IACC).

5. Zhi Zhou, Gonzalo R. Arce and Giovanni Di Crescenzo. ”Halftone Visual Cryptography”. IEEE transactions on image processing, vol. 15, no. 8, august 2006.

6. Parakh and S.kak .”A Recursive Threshold Visual Cryptography Scheme ”. Department of Computer Science, Oklahoma State University Stillwater, OK 74078.