

**R AVANEESH**

**NOVEMBER 2022**

## **ABSTRACT**

In the modern era, biometrics has emerged as a viable means of identity verification. It has been used in handheld devices like smartphones to provide easy access to the device for the user while providing good amount of confidence in ensuring that unauthorized access is not possible. Biometric authentication can be used as a secure means of identify verification for access controlled systems. In such systems, the security of the stored credentials is of utmost importance. Visual cryptography provides an easy and secure method of storing such biometric credentials. In this project, we have aimed to securely encrypt fingerprint data which can be quickly decrypted if and only if all the shares are possessed by the decrypting machine. Whereas unauthorized access is greatly minimized. We have also implemented LSB watermarking to hide one of the shares which is possessed by the user in plain sight as it is being transmitted to the decrypting device.

# TABLE OF CONTENTS

|   |           |
|---|-----------|
| List of Figures                                   | iii       |
| List of Abbreviations                             | iv        |
| <b>1 INTRODUCTION</b>                             | <b>1</b>  |
| 1.1 Cryptography . . . . .                        | 1         |
| 1.2 Confidentiality . . . . .                     | 2         |
| 1.3 Authentication . . . . .                      | 2         |
| <b>2 PURPOSE OF THE PROJECT</b>                   | <b>4</b>  |
| 2.1 Motivation . . . . .                          | 4         |
| <b>3 DESIGN</b>                                   | <b>5</b>  |
| 3.1 Different image encryption schemes . . . . .  | 5         |
| 3.1.1 Data Encryption Standard (DES) . . . . .    | 5         |
| 3.1.2 Rivest–Shamir–Adleman (RSA) . . . . .       | 6         |
| 3.1.3 Visual Cryptography System (VCS) . . . . .  | 6         |
| 3.2 Design of System . . . . .                    | 7         |
| 3.3 Visual Cryptography . . . . .                 | 9         |
| 3.3.1 Why VCS is used? . . . . .                  | 9         |
| 3.3.2 Implementation . . . . .                    | 10        |
| 3.3.3 Design of VCS . . . . .                     | 11        |
| 3.3.4 Algorithm . . . . .                         | 11        |
| 3.4 LSB Watermarking . . . . .                    | 12        |
| 3.4.1 Need for Watermarking . . . . .             | 12        |
| 3.4.2 Why is LSB Watermarking chosen . . . . .    | 13        |
| 3.4.3 Design of LSB Watermarking Scheme . . . . . | 13        |
| 3.4.4 Algorithm . . . . .                         | 13        |
| <b>4 ANALYSIS</b>                                 | <b>15</b> |
| 4.1 LSB Watermarking . . . . .                    | 15        |
| 4.2 VCS . . . . .                                 | 15        |
| <b>5 MATLAB CODE AND RESULTS</b>                  | <b>16</b> |

|          |  |           |
|----------|--|-----------|
| 5.1      | Encryption - Using LSB . . . . .           | 16        |
| 5.2      | Decryption - Using LSB . . . . .           | 23        |
| 5.3      | Using Modified LSB with VCS . . . . .      | 28        |
| 5.3.1    | Encryption . . . . .                       | 28        |
| 5.3.2    | Decryption . . . . .                       | 31        |
| 5.4      | VCS, LSB and Randomized Flipping . . . . . | 31        |
| 5.4.1    | Encryption . . . . .                       | 31        |
| 5.4.2    | Decryption . . . . .                       | 33        |
| 5.4.3    | Comparison of Run time . . . . .           | 34        |
| 5.5      | Probabilistic Nature of VCS . . . . .      | 34        |
| <b>6</b> | <b>CONCLUSION AND FUTURE SCOPE</b>         | <b>42</b> |
| 6.1      | Conclusion . . . . .                       | 42        |
| 6.2      | Future Scope . . . . .                     | 42        |
| <b>7</b> | <b>APPENDIX</b>                            | <b>43</b> |
| 7.1      | Matlab Code . . . . .                      | 43        |
| 7.1.1    | Encryption Using Modified Scheme . . . . . | 43        |
| 7.1.2    | Decryption Using Modified Scheme . . . . . | 47        |
| 7.2      | Plagiarism Report . . . . .                | 50        |
|          | <b>REFERENCES</b>                          | <b>51</b> |

## LIST OF FIGURES

|     |   |    |
|-----|---|----|
| 3.1 | DES Structure . . . . .                                 | 5  |
| 3.2 | VCS Encryption System . . . . .                         | 7  |
| 3.3 | Enrollment . . . . .                                    | 8  |
| 3.4 | Authentication . . . . .                                | 8  |
| 3.5 | Types of Shares . . . . .                               | 10 |
| 3.6 | (2,2) VCS scheme with 2 subpixel construction . . . . . | 11 |
| 5.1 | Shares Generated . . . . .                              | 30 |
| 5.2 | Encrypted output Modified LSB scheme . . . . .          | 30 |
| 5.3 | Decrypted output . . . . .                              | 31 |
| 5.4 | Output . . . . .  | 33 |
| 5.5 | Decrypted output . . . . .                              | 34 |
| 5.6 | Comparison of Run times . . . . .                       | 34 |
| 7.1 | Plagiarism Report . . . . .                             | 50 |

## **LIST OF ABBREVIATIONS**

|      |  |
|------|--|
| DES  | Data Encryption Standard                       |
| LSB  | Least Significant bit                          |
| NIST | National Institute of Standards and Technology |
| RNG  | Random Number Generator                        |
| VCS  | Visual Cryptography Scheme                     |

# CHAPTER 1

## INTRODUCTION

### 1.1 Cryptography

Cryptography was born from the need to exchange information securely between individuals. It is the study and practise of techniques to securely send confidential data from one source to another. As the communication would most often be carried out in the presence of adversarial behaviour, malicious attackers could gain access to the data transmitted. Cryptography involves techniques to ensure that the data transmitted can only be read by the indented receiver. It accomplishes this by mainly relying on various codes and algorithms to encode the data before transmission. Only the indented recipient would have access to the proper decoding scheme and can retrieve the data back. Unintended recipients may be able to receive the data but will not be to decode the data easily. Two common terms used in cryptography are plaintext and ciphertext. Plaintext is the original message which is human readable while ciphertext is a form of meaningless text that was created by applying an algorithm or a set of mathematical operations on the plaintext. The data transmitted to the receiver is the ciphertext. Intended recipients can perform the proper decoding on the ciphertext to get back the plaintext.

Modern cryptography resides at the nexus of mathematics, computer science, information security, electrical engineering, digital signal processing, physics, and other fields. Modern cryptography algorithms are based on difficult mathematical problems that are hard to solve. This in turn ensures that the algorithm is hard to break. The security of these algorithms reside in the difficulty in solving the corresponding mathematical problem. While it can be argued that even well designed cryptosystems can be broken into, it is very difficult to do so in actual practice. Such systems are therefore called 'Computationally Secure' systems. However, these systems might have to be redesigned, adapted or even changed entirely over the course of many years due to theoretical advancements towards the solution of the underlying mathematical problem as well as technical advancements resulting in faster computing power. It must be noted that, even though there are information-theoretically secure schemes which have been proved to be unbreakable even with unlimited computing power (one-time pad for example), they are not as popular as theoretically breakable, but computationally secure, schemes due to the difficulty in implementing the former.

## 1.2 Confidentiality

Confidentiality is one of the major elements in any cryptosystem. It is used to ensure that only the intended recipient can gain access to the confidential information. Even if unintended recipients are able to acquire the data transmitted by methods of eavesdropping, they should not be able to understand the information present in it. This can be done by encrypting the information using a suitable algorithm and key and transmitting the encrypted message. The intended recipient can then decrypt the message with the same key. This ensures that unintended recipients cannot decrypt the data as they do not have access to the key. However, if anyone manages to steal the key, they can naturally decrypt the data. Hence the security lies keeping the key secure. This method of using a single key for encryption and decryption is called the 'Private-Key' approach. The main crux of this approach is in finding a way to securely communicate the chosen key to the intended recipient because even the intended recipient will not be able to decrypt the data correctly without the correct key. The key used should also be periodically updated as malicious attackers might be able to deduce the key through various crypto-analysis methods after a sufficiently long period of time. This means that every time the key is updated, it would have to be shared with all the intended recipients. This by itself possess huge security risks. The key could be stolen in one of the many sharing processes or it could be stolen from one of the many key holders.

An alternative to this system is for each user to have his/her own 'Public-Key' and 'Private-Key'. Encryption of messages would be done using the public keys while the decryption would be done using the private keys. Every users' public keys would be made available to every other user while the private keys would never be shared with anyone. In this system, if a user (Alice) wants to share information securely to another user (Bob), Alice would encrypt the data with Bob's public key and transmit it. Bob would then receive the data and decrypt it with his own private key. As long as Bob keeps his private key secure no other user would be able to decrypt the data immediately. This system also requires the keys to be updated periodically, however the advantage is that the communication of the public keys do not have to be secure. Every user can update his/her own public and private keys periodically and simply broadcast the public key. Even if a malicious attacker manages to acquires the users' public keys, they would not be able to use it for decryption. Hence the second method is generally preferred over the first method due to its ease of implementation.

## 1.3 Authentication

Authorisation is a component of any cryptosystem by which the sender and receiver can confirm each others identities during data transmission. Consider a case where Alice wants to communicate with Bob using a public key cryptosystem and Eve wants to eavesdrop on their communication. Although the encryption itself would be hard to break as Eve will not have access to Bob's private key, Eve can still executes a



'Man-in-the-Middle" attack to trick Alice into using Eve's public key to encrypt the data instead of Bob's. Now, only Eve will be able to decrypt the data. Not even Bob, the intended recipient would be able to do so. After receiving the data, Eve can then transmit a slightly altered or a completely different data to Bob. Such an attack would raise concerns about the integrity and the authenticity of the message. This would mean that neither Alice nor Bob would realise that they are communicating with Eve instead of each other.

Such attacks can be resisted against by the use of a digital signature which is a mathematical technique used to validate the authenticity and integrity of a message. Digital signature works by relying on the mutually authenticating feature of cryptographic keys. Consider a case where Alice wants to send a message to Bob. To ensure confidentiality, she would encrypt the message with Bob's public key so that he can decrypt it using his own private key. The digital signature related data would also be tacked onto the data transmission. However the digital signature related data would be encrypted using Alice's private key and Bob would try to decode it using Alice's public key. If the signature related data could not be properly decoded, that means that either the data did not originate from Alice or the data was tampered with. The digital signature can only perform its functions properly if the signee has kept his/her own private key secure. If a user's private key was stolen, malicious attackers can then impersonate the user by creating a digital signature from the private key.

## **CHAPTER 2**

### **PURPOSE OF THE PROJECT**

In the modern world, fingerprints are one of the most commonly used method for identity verification. They are more convenient to use than passwords. Most handheld devices like smartphones are equipped with a fingerprint scanner to perform easy biometric authentication to grant users safe access to their devices. Fingerprints are also used by many organisation to safe guard their data as well as to restrict access to certain locations. However, to perform fingerprint authentication, the fingerprints would first have to be stored in a company database. The security of the database and the data itself has to be guaranteed. This is because once the fingerprints are leaked, unauthorized personal can use these 'authorized' fingerprints to gain access to confidential data/locations previously secured by them. Fingerprints being leaked are a much higher cause for concern than passwords being leaked because passwords can be changed while fingerprints cannot. For this reason, fingerprints are usually encrypted before they are stored in databases.

#### **2.1 Motivation**

Consider a scenario where a confidential location of a company needs fingerprint authentication for access. One method to implement this would be store the fingerprints of all the authorized personal into the company database and use a fingerprint scanner at the entrance. Personal would first have to get their fingerprints scanned and would only be allowed access if the fingerprint scanned matches with one in the database. One inherent flaw in this system that the fingerprints stored in the database can be leaked. Even if the fingerprints are encrypted before storage, the attacker might still be able to decode the correct fingerprint or atleast parts of it, if given enough time and computational power. Another concern is the computational power required for the decryption. The decryption would be done at the fingerprint scanner, which are lightweight, compact devices. They would not be able to handle a decryption scheme which requires huge amounts of computational power. Therefore, the purpose of the study is to create a system which ensures that a proper fingerprint cannot be decoded even if the data stored in the database was compromised, at the same time ensuring a low computational power requirement for decryption.

## CHAPTER 3

### DESIGN

#### 3.1 Different image encryption schemes

##### 3.1.1 Data Encryption Standard (DES)

Data Encryption Standard (DES) was once the most dominant cryptographic schemes available. It was first introduced by the National Institute of Standards and Technology (NIST) and uses a symmetric-key block cipher. DES is implemented using a Feistel Block Cipher consisting of 16 rounds. DES uses a block size of 64 bits. It also has a key length of 64 bits, however only 56 bits are used for the encryption process. Hence the effective key length of DES is 56 bits. The rest 8 bits are used as check bits.

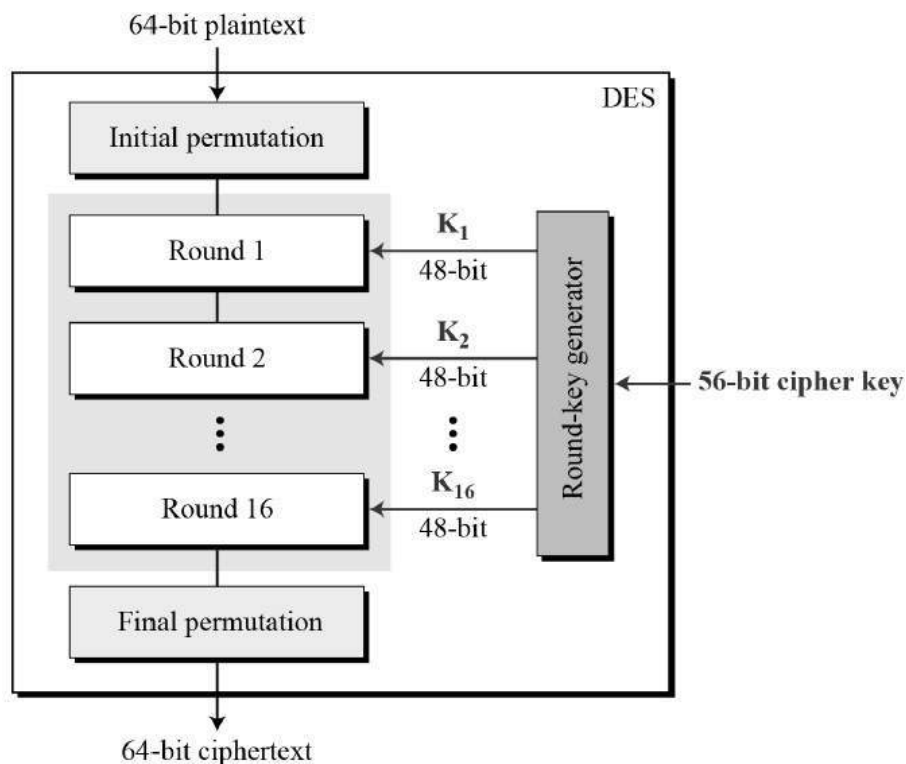


Figure 3.1: DES Structure

First step of DES is Initial Permutation where the 64 bits of the input are permuted. This permuted output is then given as input to the first round of 16 rounds of operation. Each of these rounds involves permutation and substitution and has a different 48-bit round key produced from the cipher key according to a predefined

algorithm. The final step of DES is the 'Final Permutation' which takes the output of the previous 16th round as input and the 64 bit ciphertext as the output. Final Permutation is defined as the inverse of the Initial Permutation.

Over the years, many security flaws in DES were exposed and cryptographers were forced to either improve upon the existing DES or develop a new algorithm entirely.

### **3.1.2 Rivest–Shamir–Adleman (RSA)**

RSA is an asymmetric cryptographic algorithm, meaning that two different keys are used for encryption and decryption namely, a Public Key and a Private Key. An RSA user will create the set of keys using two large prime numbers, after which the Private Key and the two prime numbers are kept secure by the user while the Public Key is made available to the public. Anyone can use the user's public key to encrypt a message while only someone with access to Private Key can easily decrypt it. RSA belongs to the class of "theoretically breakable but computationally secure" cryptosystems. Its security lies in the practical difficulty of prime factorising the product of two large prime numbers. RSA offers a large degree of security, but at the same time suffers from the disadvantage of being very slow. Hence, RSA is very rarely used to encrypt the actual data itself. Most cryptosystems implement a hybrid system, where the actual data is encrypted using a Private Key encryption scheme and this Private Key is encrypted using RSA and shared with the intended recipient. This speeds up the encryption/decryption process while also mitigating the risk of the key being stolen.

While RSA is quite secure it is also computationally complex and hence is difficult to implement using devices with computational capabilities and low power requirements.

### **3.1.3 Visual Cryptography System (VCS)**

Visual cryptography involves the encryption of visual data like images, into two or more components or shares. The key point in VCS is that the data can be decoded using sight reading. The shares generated will be semi transparent and overlaying the shares on top of each other will reveal the actual image. It is impossible to decode the image correctly without all the shares. Each pixel of the image is subdivided into four pixels and filled in the following manner. If the original pixel in the image is set (black), we fill in all four sub pixels then distribute them two per cipher layer. The question of which pattern is placed on which layer is decided randomly. Each has a 50% chance of occurring. It does not matter which pair of pixels goes on which layer, when they are combined, all four pixels will be black. Conversely, if the source image pixel is white, then only two of the four subpixels are shaded/set. This time, however, the same two pixels are shaded on both layers. This means that, when the two cipher images are combined, only two pixels will be shaded. The chiral set to be used is determined by probability. Each has a 50% chance of occurring. Decoding can be done easily by XORing each pixel value in a share with the correspondingly pixel value in the other share.

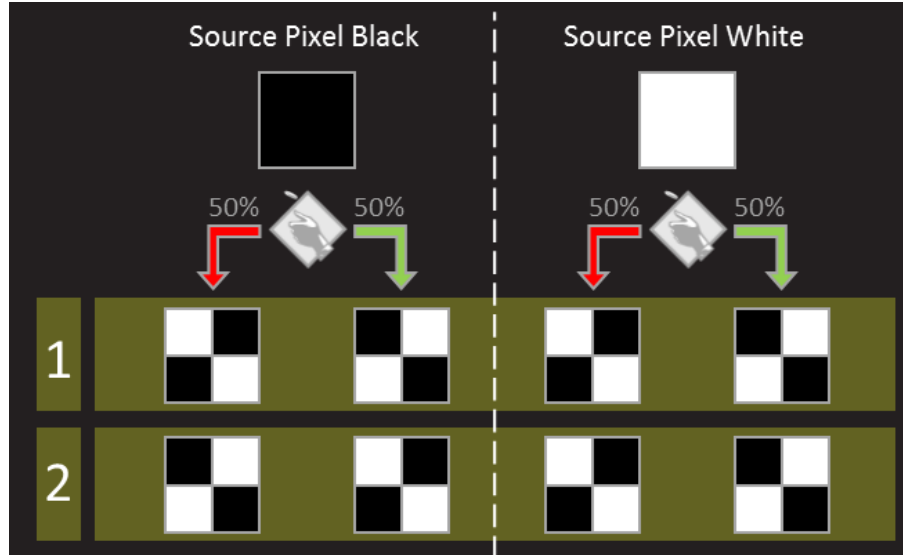


Figure 3.2: VCS Encryption System

### 3.2 Design of System

The proposed system uses VCS to encrypt the fingerprints of the employees. Fingerprints are first obtained from the employees and converted into binary data. VCS is performed on this binary data to produce two shares. One of the shares will be stored in the company database and the other share will be embedded into a photograph of the employee using LSB watermarking. There are two different types of LSB watermarking. The first method involves normal LSB watermarking technique, whereas in the other the secret image is flipped left to right before LSB watermarking is performed. We use a combination of both of these in which a particular row of the secret image (share) may or may not be flipped with a probability of 0.5. The employee id is used as the seed for a random number generator based on whose outputs, the rows of the secret image are flipped. The picture used will be an RGB image and the decision of which layer to perform the LSB watermarking would depend on the employee id. The employee id reduced mod 3 will indicate which layer is to be used. 0 indicates Red layer, 1 indicates Green layer and 2 indicates Blue layer. This picture along with other employee information like name, employee id, department etc will be stored on to the employee's company issued smartphone. The smartphone would be performing the function of a key card. The employee can scan the smartphone on an RFID/NFC scanner which would then transmit the employee data stored in the smartphone to the scanner. This can be used for providing a lower level of authentication for access to less important company data and locations. The only requirement to access these data and locations is to prove that the personal is actually an employee of the company.

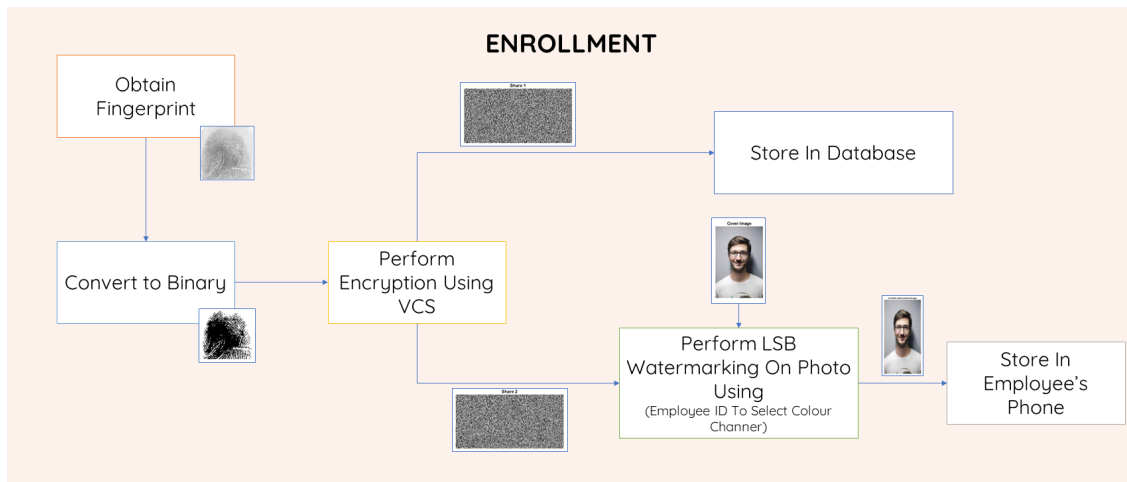


Figure 3.3: Enrollment

For access to more important company data and locations, a fingerprint as well as smartphone authentication is required. After receiving the employee data stored in the employee's smartphones, the scanner would first pull the finger print share corresponding to the employee id stored in the company database. The employee id reduced mod 3 will correspond to which layer of the image LSB watermarking was performed on. After extracting that particular layer from the image, an extraction algorithm is used to extract the second share embedded into the picture. With both shares now available, they would be decrypted using simple XOR operations and result would be validated with the actual fingerprint inputted by the employee. Validation is performed by taking *xor* of decrypted image matrix and scanned fingerprint matrix, to determine the amount of pixel/bit mismatches. If this mismatch is less than a threshold (10%), we consider the fingerprints to be a match and the user is authenticated.

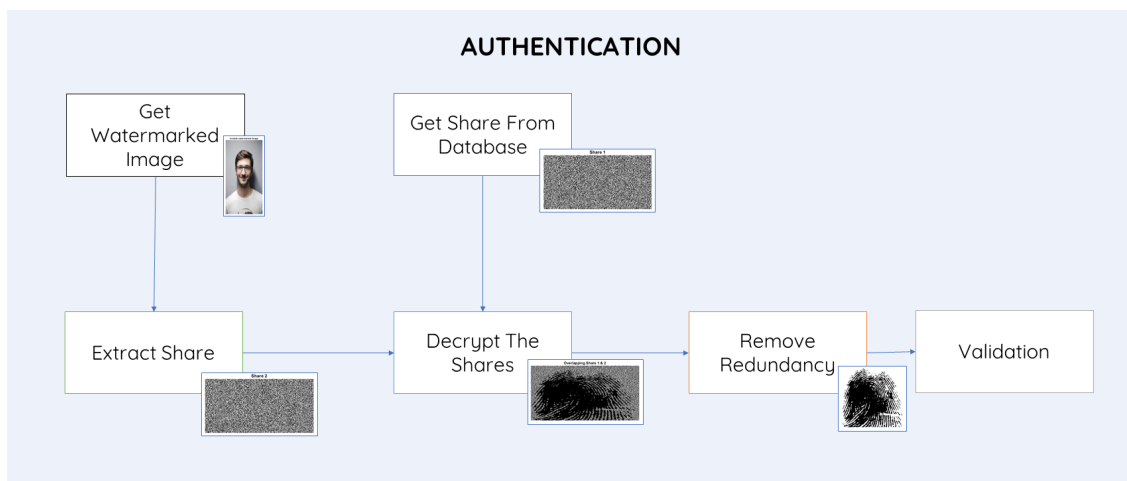


Figure 3.4: Authentication

### 3.3 Visual Cryptography

Naor and Shamir [1] introduced the visual cryptography scheme (VCS) as a simple and secure way to allow the secret sharing of images in such a way that simple superimposition can be used for decryption. The fundamental component of the system is cipher text that is transmitted to the intended user via wireless media. The key is one of the split plain text. The original picture/plain text can be seen when the key is put over the encryption text. The simplicity of this approach is its distinguishing quality.

#### 3.3.1 Why VCS is used?

- Decryption is simple and does not involve any cryptographic computations.
- Encryption is simple to perform and does not take much time or computational resources.
- It possesses cipher text indistinguishability as the same message is not always mapped to the exact cipher text, i.e. it is semantically secure. This is because it is probabilistic in nature.
- It is secure as discussed below.

#### Information Theoretic Secrecy

If a cryptosystem is secure against adversaries with unlimited computing resources and time, it is said to have information-theoretic security (also known as unconditional security). Protocols proven to be information-theoretically secure are resistant to future advances in computing. Entropy-wise, knowledge of the cipher text has no effect on the entropy of plain text i.e knowledge of cipher text does no help in determining plaintext. [2].

$$H(P | C) = H(P) \quad (3.1)$$

where  $H(P | C)$  is entropy of plain text given cipher text and  $H(P)$  is entropy of plain text

This implies

$$H(K) \geq H(P) \quad (3.2)$$

where  $H(K)$  is entropy of key and  $H(P)$  is entropy of plain text

or that the length of the key should be greater than or equal to that of the message.

VCS is secure as it follows Information Theoretic Secrecy. Here, the size of the shares is equal to or greater than the size of the plain text image. (Specifically in (2,2) VCS it is double that of plain text).

## Secret Sharing

It refers to methods of disseminating a secret among a group in such a way that no individual has any discernible knowledge of the secret. However, if a large enough number of people pool their "shares," the secret can be reconstructed. While insecure secret sharing enables an attacker to gain additional information with each share, secure secret sharing is 'all or nothing,' which means that anyone with fewer than 't' shares knows no more about the secret than someone with 0 shares [2]. VSC is safe because owning one share gives no information about the other. There is no computational way to obtain the other share, as information regarding it does not exist in the first share. The only way to obtain the other shares is by guessing all possibilities.

### 3.3.2 Implementation

Given a raw binary image, it is encrypted in such a way that:

$$T = S_{h1} \oplus S_{h2} \oplus \dots \oplus S_{hk} \quad (3.3)$$

Where  $S_{hi}$  are the shares that appear as random noise individually, ( $k \leq n$ ), the encryption is done in such a manner that  $k$  or more of the generated images are required for reconstructing the original image.

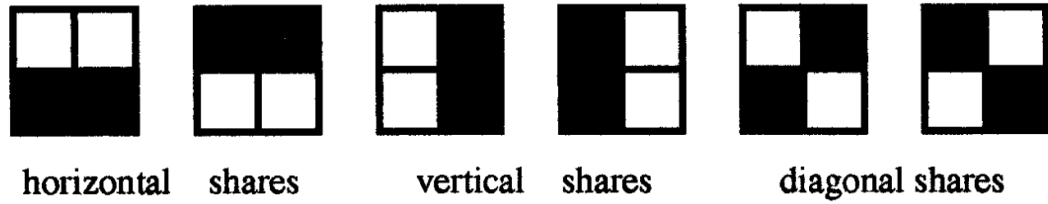


Figure 3.5: Types of Shares

Every pixel in the original image is encrypted into two subpixels called shares in the case of (2, 2) VCS. The distribution of shares for a white and black pixel is determined at random. Because different pixels in the secret image will be encrypted using independent random choices, neither share provides any information about the original pixel. [3].

The value of the original pixel can be determined by superimposing the two shares. If the pixel is black, we get two black subpixels; if the pixel is white, we get one black subpixel and one white subpixel. As a result, the reconstructed image will be twice as wide as the original secret image, with a 50% loss in contrast. This can be avoided by choosing one out of every two subpixels. In a binary image, 0 represents black and 1 represents white. A black subpixel is [00] and a white subpixel is [01]; by taking the max() of two adjacent subpixels, we can reduce the decrypted image to the size of the secret image while also improving contrast. [4].





















| PIXEL   | SHARES  |   | SUPERPOSITION OF TWO SHARES   |   |                |
|---|---|---|---|---|----------------|
|   | #1  | #2  | #OR   | #XOR  |                |
|  |  |  |  |  | WHITE<br>PIXEL |
|   |  |  |  |  |                |
|  |  |  |  |  | BLACK<br>PIXEL |
|   |  |  |  |  |                |

Figure 3.6: (2,2) VCS scheme with 2 subpixel construction

### 3.3.3 Design of VCS

#### Deciding the Number of Shares

Here, we have taken (2,2) VCS. We have chosen  $k = 2$  and  $n = 2$  as one share will be with the user and the other with the decrypting device, so we need only two shares, both of which must be used to decrypt the fingerprint image. We may have taken a higher value of  $n$ . However, as the shares are to be kept with only two parties, and in light of the fact that many IoT devices used for decryption have limited memory, it is best to go for the scheme requiring less memory. (2,2) VCS is also simple to encode and decode as only two shares are involved. As (2,2) VCS also provides adequate security, it is chosen.

### 3.3.4 Algorithm

#### Encryption Algorithm

1. Select Secret image
2. In binary images as 0 is black and 1 white, Let the subpixels be represented as two row vectors
  - $s1a = [1\ 0]$  and  $s1b = [1\ 0]$  for white pixel
  - $s0a = [1\ 0]$  and  $s0b = [0\ 1]$  for black pixel
3. Generate the shares
  - For each pixel of input:

- Choose 0 or 1 randomly ( $p = \frac{1}{2}$ )
- if 0 is chosen Shares are based on subpixels given above
- Else if 1 is chosen, new subpixels are [0 1] [0 1] for white and [0 1] [1 0] for black.
- Store s1a/s0a in share 1 and s1b/s0b in share 2.

4. The two share have now been generated

### **Decryption Algorithm**

1. Perform bitwise 'or' operation on share 1 and share 2
2. Negate the output
3. Remove redundancy
  - For every two pixels in recovered image, take the one with maximum value to obtain original image back
4. The image is now decrypted

## **3.4 LSB Watermarking**

Steganography is the practice of hiding information within another message or physical object so that human inspection does not reveal its presence. In computing/electronic contexts, a computer file, message, image, or video is hidden within another file, message, image, or video.

LSB Watermarking, is a means for concealing an image("Secret image") in another image("cover image"). It hides the binary secret image into the least significant bit of the cover image [5].

A variant of LSB watermarking includes flipping the secret image before embedding it into the cover image. We use a combination of the normal LSB watermarking as well as the variant to randomly determine if a particular row of the secret image is to be flipped or not prior to embedding into cover image. For the same, we use a random number generator for which the employee id is taken as seed.

### **3.4.1 Need for Watermarking**

The user's share must be transmitted from the user to the verification device. Though the share is encrypted, it is unwise to broadcast that we have a secret to hide. In such cases, it is better if any eavesdropper does not even realise that a secret is being shared. LSB watermarking can help us hide the share in the user's photograph. Thus any eavesdropper only sees an innocent photo of the user being transmitted [6].

### 3.4.2 Why is LSB Watermarking chosen

- It is easy to implement.
- Inverse watermarking is also easy to implement
- It has high fidelity.
- The MSE of extracted secret image is low (low distortion).
- The impact of watermark is negligible on the cover object.

*Note: LSB watermarking does not provide much security as it merely hides the secret image.*

### 3.4.3 Design of LSB Watermarking Scheme

#### 3.4.4 Algorithm

##### Embedding Algorithm

1. Read the cover image
2. Read the watermark image to hide in the cover image
3. Determine the dimensions of the cover image and the watermark.
4. For each row of the watermark
  - By means of RNG generate a random number 0 or 1.
  - If 1 is generated, flip the row of watermark.
5. Set the LSB of each pixel in the cover image to the value of the modified watermark pixels..
6. Generate the watermarked image.

##### Extraction Algorithm

1. Read in watermarked image
2. Determine the size of watermarked image
3. Use LSB of watermarked image to recover the modified watermark.
4. For each row of modified watermark
  - Using the same seed used for embedding, generate random numbers 0 or 1 by means of RNG.

- If 1 is generated, flip the row of modified watermark.

5. Scale and display recovered watermark

## CHAPTER 4

### ANALYSIS

#### 4.1 LSB Watermarking

LSB watermarking offers no cryptographic security, while it does offer steganographic security by hiding the encrypted information (VCS share 2) inside an innocent cover image (face image). However, as the attacker does not know which layer of the cover image has been used for LSB watermarking, he/she will have to try extracting the share from all three layers and try to decrypt them as it is not possible to tell which of the extracted matrix of zeros and ones is the actual share as the share itself looks like random noise. We have used the last bit for LSB watermarking as extracting the share from the cover is relatively simple and extracted share has high fidelity and low PSNR.

#### 4.2 VCS

As VCS is information theoretically secure, (i.e. it is a visual form of secret sharing) being in possession of less than 2 shares is equivalent to having possession of zero shares, as even though the attacker possesses one share, since the information about the other share cannot be derived from the possessed share. Thus the only choice left to the attacker is to guess all possible combinations available for the other share. This means that in our case, the attacker needs to guess correctly for one share out of  $2^{300 \times 600}$ .

As each pixel of the share may be either black or white, each pixel has 2 possible values. Like this the attacker has to guess correctly for  $300 \times 600$  pixels as generation of the shares leads to expansion of dimensions. (in our case of the width by a factor of 2).

When the attacker intercepts the face image while it is being transmitted, he/she needs to first identify which of the 3 layers contains the share and then needs to guess the other share correctly.

The attacker would then need to determine if a row has been flipped or not. There are two possible choices for each row and we have 600 rows. So in all we have  $3 * 2^{600} * 2^{300 \times 600}$  available options for the attacker to obtain first share from the cover image, deduce which rows have been flipped and subsequently deduce the other share as well.

## CHAPTER 5

### MATLAB CODE AND RESULTS

#### 5.1 Encryption - Using LSB

##### Image Encryption using VCS and LSB Watermarking

```
clc;  
clear;  
close all;
```

##### Visual Cryptography

```
% Read Input Secret (fingerprint) Image  
  
inImg = imread('102_3.tif');  
figure;  
imshow(inImg);title('Secret Image');
```

Secret Image



```
% Convert Secret Image to Binary  
  
inImg = imbinarize(inImg);  
figure;
```

```
imshow(inImg);title('Binary Secret Image');
```

**Binary Secret Image**



```
% Perform Visual Cryptography
```

```
[share1, share2] = VisualCryptography(inImg);
```

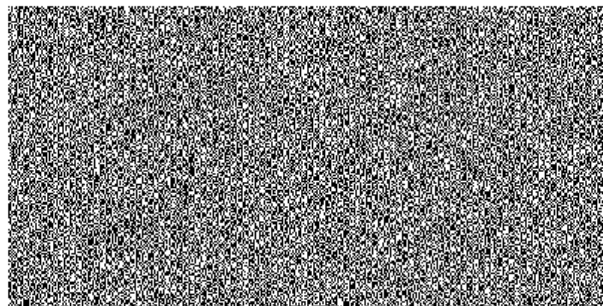
```
White Pixel Processing...
```

```
Black Pixel Processing...
```

```
Share Generation Completed.
```

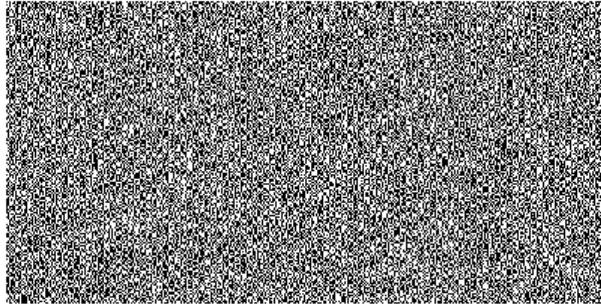
```
figure;imshow(share1);title('Share 1');
```

**Share 1**



```
figure;imshow(share2);title('Share 2');
```

**Share 2**



```
% Save the shares  
disp('Saving.....')
```

Saving.....

```
imwrite(share1, 'Share1.tif');  
imwrite(share2, 'Share2.tif');  
disp('Done')
```

Done

## LSB Watermarking

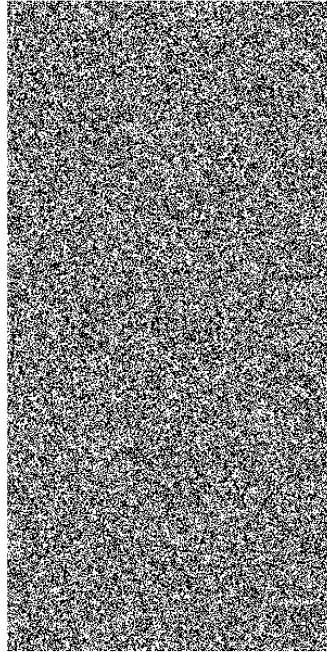
```
% LSB Watermarking of Share 2  
disp('Performing LSB watermarking')
```

Performing LSB watermarking

```
employee = 123456789;  
  
% reshape to portrait  
s = size(share2);  
share2 = reshape(share2, [s(2), s(1)]);  
share2 = imbinarize(share2);  
figure; imshow(share2); title('Share 2 Rotated');
```



**Share 2 Rotated**



```
% Get Photo and resize it  
face = imread('face.jpeg');  
s = size(share2);  
figure;imshow(face);title('Cover Image');
```

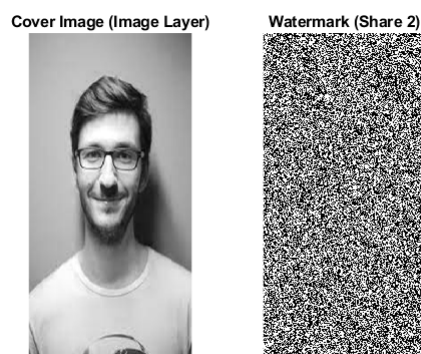
**Cover Image**



```
face = imresize(face,s);
figure;imshow(face);title('Resized Cover Image');
```



```
% Get the layer in which share will be hidden
x = rem(employee,3)+1;
layer = face(:, :, x);
figure;subplot(121);imshow(layer);
    title('Cover Image (Image Layer)');
    subplot(122);imshow(share2);
    title('Watermark (Share 2)');
```



```
% least significant bit substitution
z=double(share2);
r=double(layer-mod(layer,2)); % removal of LSB bits
```

```
l=uint8(r+z);

face(:,:,x) = l;
figure;imshow(face);title('Invisble watermarked Image');
```



```
disp('Lsb wartermarking done')
```

Lsb wartermarking done

```
% Saving watermarked image
disp('Saving watermarked image')
```

Saving watermarked image

```
imwrite(face,'face.tif');
disp('Done')
```

Done

```
function [share1, share2] = VisualCryptography(inImg)
s = size(inImg);
share1 = zeros(s(1), (2 * s(2)));
share2 = zeros(s(1), (2 * s(2)));

% White Pixel Processing
```

```

disp('White Pixel Processing...');
% White Pixel share combinations

sla=[1 0];
slb=[1 0];
[x , y] = find(inImg == 1);
len = length(x);
for i=1:len
    a=x(i);b=y(i);
    pixShare=ShareGenerator(sla,slb);
    share1((a),(2*b-1):(2*b))=pixShare(1,1:2);
    share2((a),(2*b-1):(2*b))=pixShare(2,1:2);
end
% Black Pixel Processing
disp('Black Pixel Processing...');
% Black Pixel share combinations

s0a=[1 0];
s0b=[0 1];
[x ,y] = find(inImg == 0);
len = length(x);
for i=1:len
    a=x(i);b=y(i);
    pixShare=ShareGenerator(s0a,s0b);
    share1((a),(2*b-1):(2*b))=pixShare(1,1:2);
    share2((a),(2*b-1):(2*b))=pixShare(2,1:2);
end
disp('Share Generation Completed.');
```

```

end

function out = ShareGenerator(a,b)
a1 = a(1);
a2 = a(2);
b1 = b(1);
b2 = b(2);
in = [a
```

```

        b];
out = zeros(size(in));
randNumber = randi([0,1]);
if (randNumber == 0)
    out = in;
elseif (randNumber == 1)
    a(1) = a2;
    a(2) = a1;
    b(1) = b2;
    b(2) = b1;
    out = [a
           b];
end
end
end

```

## 5.2 Decryption - Using LSB

### Decryption

```

clc;
clear;
close all;

```

```

% Read the images
face = imread('face.tif');
share1 = imread('Share1.tif');
share1 = imbinarize(share1);

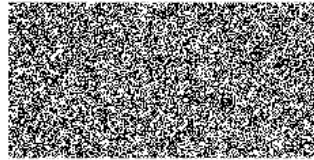
figure;subplot(121);imshow(face);
        title('Cover Image containing Share 2');
        subplot(122);imshow(share1);title('Share 1');

```

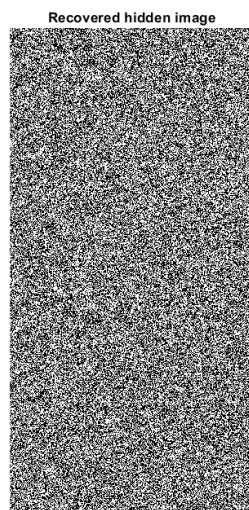
Cover Image containing Share 2



Share 1

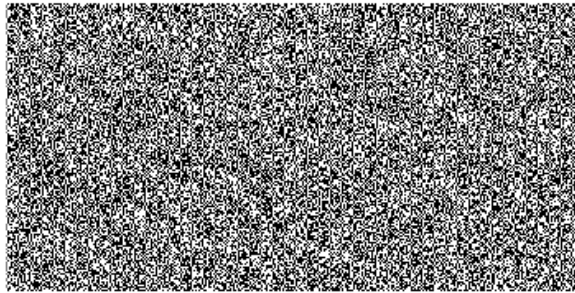


```
employee = 123456789;  
x = rem(employee,3)+1;  
% inverse watermarking  
  
layer = face(:, :, x);  
a=mod(layer,2);  
b=zeros(600,300);  
for x=1:600  
    for y=1:300  
        if(a(x,y)==1)  
            b(x,y)=255;  
        end  
    end  
end  
recImg=imbinarize(b);  
figure  
imshow(recImg); % hidden image  
title('Recovered hidden image')
```

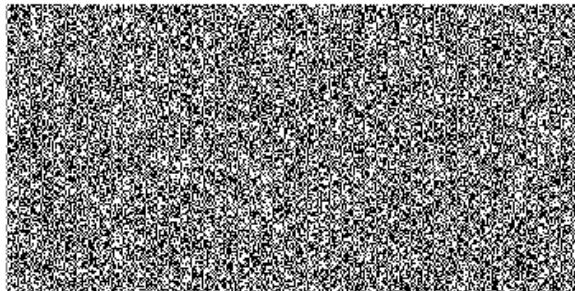


```
s = size(recImg);  
share2 = reshape(recImg, [s(2), s(1)]);  
  
figure; subplot(211); imshow(share1); title('Share 1');  
        subplot(212); imshow(share2); title('Share 2');
```

**Share 1**



**Share 2**



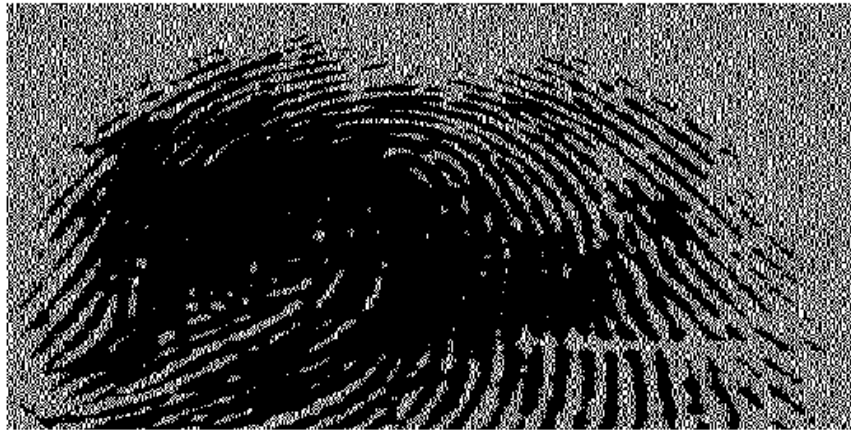
```
% Decrypting the shares

share12 = bitor(share1, share2);
share12 = ~share12;

figure;imshow(share12);title('Overlapping Share 1 & 2');
```



Overlapping Share 1 & 2



```
s = [300 , 300];  
shareop = zeros(s);  
for i = 1:s(1)  
    for j = 1:s(2)  
        a = share12(i, (2*j-1):(2*j));  
        shareop(i,j) = max(a);  
    end  
end  
  
figure;imshow(shareop);title('Decrypted output');
```

Share op



```
inImg = imread('102_3.tif');
inImg = imbinarize(inImg);
figure;imshow(inImg);title('Secret Image');
```



```
thresh = xor(shareop,inImg);
thresh = sum(sum(thresh));
if thresh < 0.1*size(inImg,1)*size(inImg,2)
    disp("Authenticated.")
end
```

Authenticated

## 5.3 Using Modified LSB with VCS

### 5.3.1 Encryption

In this case, before we perform LSB Watermarking, the secret image (Share 2) is flipped and this flipped version is used for Watermarking.

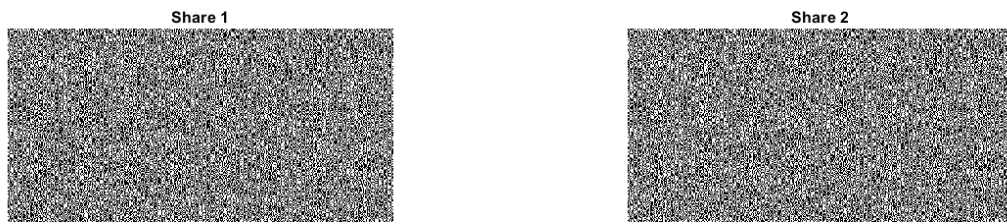
#### Code Snippet

```
share2_flip = fliplr(share2);
figure;imshow(share2_flip);title(['Fliped share 2']);

disp("fliped share")
disp(share2(1,end-5:end))
disp("share")
```

```
disp(share2_flip(1,1:6))
```

## Outputs



Share 1

Share 2

```
fliped share
  1  0  1  0  1  1
share
  1  1  0  1  0  1
```

Values of share 2 befor and after flipping

Figure 5.1: Shares Generated



Rotated and Fliped Share 2

Watermarked image

Figure 5.2: Encrypted output Modified LSB scheme

### 5.3.2 Decryption

In this case, after extracting the share 2 we flip it back before using it for decryption.

**Code:**

```
share2 = fliplr(recImg);
```

**Output**



Figure 5.3: Decrypted output

## 5.4 VCS, LSB and Randomized Flipping

In this case, we utilize employee id or any other predecided data that is transmitted by the user to the decryption machine, as a seed to a random number generator (RNG). Then we use the initialized RNG to generate a set of 600 (one for each row of rotated share 2) random bits of 0s and 1s. If a bit in position  $i$  is 1, we will flip the corresponding row  $row_i$  of share 2. In this way we are introducing confusion, diffusion and randomness.

At the receiver side, before decrypting received share, we once again use the predecided parameter as seed to regenerate the random numbers to flip the modified rows to get the original share 2 back. We then use this share and share 1 available with the decryption machine for VCS decryption, to get the fingerprint image.

### 5.4.1 Encryption

**Code**

```

% LSB1 or LSB2 randomly chosen
s = rng(employee)

a = randi([0 1],600,1);

share2_flip = zeros(600,300);
for i = 1:size(share2,1)
    if a(i) ==1
        share2_flip(i,1:end) = fliplr(share2(i,1:end));
    else
        share2_flip(i,1:end) = share2(i,1:end);
    end
end
share2_flip = imbinarize(share2_flip);
figure;imshow(share2_flip);title(['Fliped share 2']);

```

## Outputs

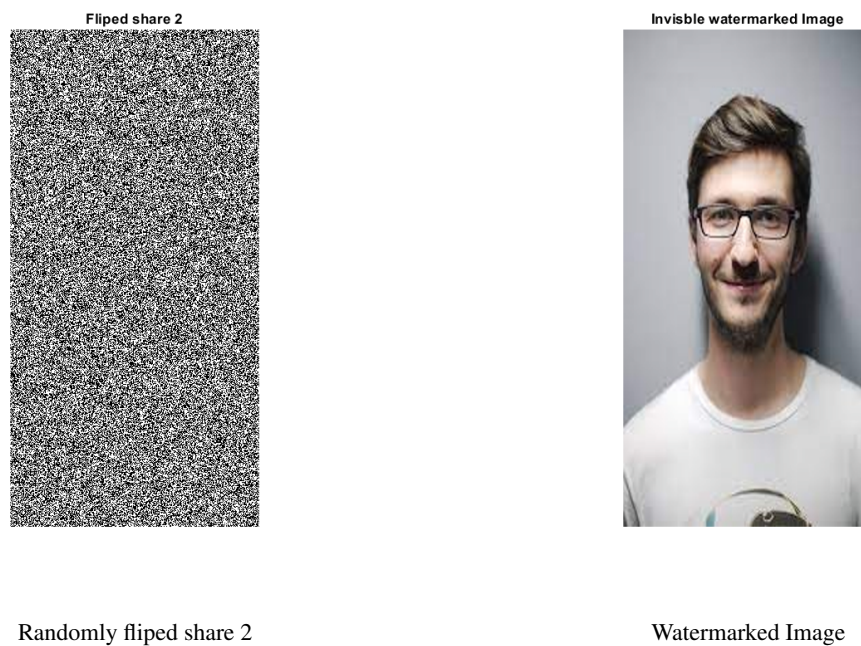


Figure 5.4: Output

### 5.4.2 Decryption

#### Code

```
s = rng(employee)

a = randi([0 1],600,1);

share2_flip = zeros(600,300);
for i = 1:size(recImg,1)
    if a(i) ==1
        share2_flip(i,1:end) = fliplr(recImg(i,1:end));
    else
        share2_flip(i,1:end) = recImg(i,1:end);
    end
end
share2 = imbinarize(share2_flip);
figure;imshow(share2_flip);title(['Fliped share 2']);
```

## Outputs



Figure 5.5: Decrypted output

### 5.4.3 Comparison of Run time

The run times for encryption and decryption using different schemes was compared using matlab.

|  |  |   |
|--|--|---|
| Time take for LSB 1 Encryption<br>0.9301 | Time take for LSB 2 Encryption<br>0.9132 | Time taken for Random FLip - Encryption<br>1.3796 |
| Encryption LSB1                          | Encryption LSB2                          | Encryption randomly LSB1 or<br>LSB2               |
| Time taken for Decryption<br>0.1651      | Time taken for Decryption<br>0.1787      | Time taken for Decryption<br>0.2437               |
| Decryption LSB1                          | Decryption LSB2                          | Decryption randomly LSB1 or<br>LSB2               |

Figure 5.6: Comparison of Run times

## 5.5 Probabilistic Nature of VCS

This shows VSC is IND-CPA



```

clc;
clear;
close all;
%Read Input Binary Secret Image
inImg = imread('102_3.tif');
figure;
imshow(inImg);title('Secret Image');

```



```

inImg = imbinarize(inImg);

figure;
imshow(inImg);title('Secret Image');

```



```

%Visual Cryptography
[share1, share2, share12] = VisCrypt(inImg);

```

```
White Pixel Processing...
Black Pixel Processing...
Share Generation Completed.
```

```
%Outputs
figure;
subplot(321)
imshow(share1);title('Share 1 - first time');
subplot(323)
imshow(share2);title('Share 2 -first time');
% figure;imshow(share12);title('Overlapping Share 1 & 2');

s =size(inImg);
shareop = zeros(size(inImg));
for i = 1:s(1)
    for j = 1:s(2)
        a = share12(i,(2*j-1):(2*j));
        shareop(i,j) = max(a);
    end
end
% imwrite(share1,'Share1.tif');
% imwrite(share2,'Share2.tif');
% imwrite(share12,'Overlapped.bmp');

subplot(325)
imshow(shareop);title('Share op 1');

%Visual Cryptography second time
[share1_2, share2_2, share12_2] = VisCrypt(inImg);
```

```
White Pixel Processing...
Black Pixel Processing...
Share Generation Completed.
```

```
%Outputs
subplot(322)
imshow(share1_2);title('Share 1 - second time');
```

```

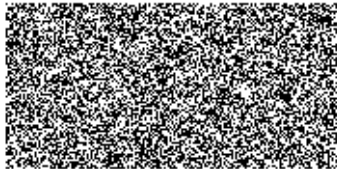
subplot(324)
imshow(share2_2);title('Share 2 -second time');
% figure;imshow(share12);title('Overlapping Share 1 & 2');

s =size(inImg);
shareop_2 = zeros(size(inImg));
for i = 1:s(1)
    for j = 1:s(2)
        a = share12_2(i,(2*j-1):(2*j));
        shareop_2(i,j) = max(a);
    end
end
end
% imwrite(share1,'Share1.tif');
% imwrite(share2,'Share2.tif');
% imwrite(share12,'Overlapped.bmp');

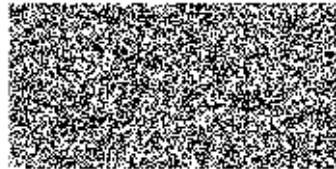
subplot(326)
imshow(shareop_2);title('Share op 2');

```

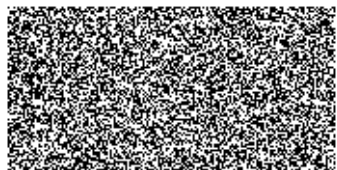
**Share 1 - first time**



**Share 1 - second time**



**Share 2 -first time**



**Share 2 -second time**



**Share op 1**



**Share op 2**



```
if shareop==inImg & shareop_2 ==inImg
    disp('Correct decoding')
end
```

Correct decoding

```
disp("To show probalilistic nature of shares generation")
```

To show probalilistic nature of shares generation

```
if share1 == share1_2
    disp("Share 1 generated first and
        second time are equal")
else
    disp("Share 1 generated first and
        second time are not equal")
end
```

Share 1 generated first and second time are not equal

```
if share2 == share2_2
    disp("Share 2 generated first and
        second time are equal")
else
    disp("Share 2 generated first and
        second time are not equal")
end
```

Share 2 generated first and second time are not equal

```
disp("To show we need the correct shares for decoding,")
```

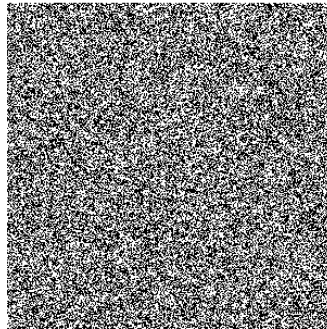
To show we need the correct shares for decoding,

```
disp("we will now show output if share 1 and  
share2_2 are used")
```

we will now show output if share 1 and share2\_2 are used

```
share12_w=bitor(share1, share2_2);  
share12_w = ~share12_w;  
shareop_w = zeros(size(inImg));  
for i = 1:s(1)  
    for j = 1:s(2)  
        a = share12_w(i, (2*j-1):(2*j));  
        shareop_w(i,j) = max(a);  
    end  
end  
figure;  
imshow(shareop_w);  
title('Share output from share1 and share2\_2 ');
```

**Share output from share1 and share2\_2**



```
if shareop_w==inImg  
    disp('Correct decoding')  
else  
    disp('InCorrect decoding')  
end
```

InCorrect decoding

```

function [share1, share2, share12] = VisCrypt(inImg)
s = size(inImg);
share1 = zeros(s(1), (2 * s(2)));
share2 = zeros(s(1), (2 * s(2)));
%%White Pixel Processing
%White Pixel share combinations
disp('White Pixel Processing...');
sla=[1 0];
slb=[1 0];
[x y] = find(inImg == 1);
len = length(x);
for i=1:len
    a=x(i);b=y(i);
    pixShare=generateShare(sla,slb);
    share1((a),(2*b-1):(2*b))=pixShare(1,1:2);
    share2((a),(2*b-1):(2*b))=pixShare(2,1:2);
end
%Black Pixel Processing
%Black Pixel share combinations
disp('Black Pixel Processing...');
s0a=[1 0];
s0b=[0 1];
[x y] = find(inImg == 0);
len = length(x);
for i=1:len
    a=x(i);b=y(i);
    pixShare=generateShare(s0a,s0b);
    share1((a),(2*b-1):(2*b))=pixShare(1,1:2);
    share2((a),(2*b-1):(2*b))=pixShare(2,1:2);
end
share12=bitxor(share1, share2);
share12 = ~share12;
disp('Share Generation Completed.');
```

```

end

function out = generateShare(a,b)

```

```

a1 = a(1);
a2 = a(2);
b1 = b(1);
b2 = b(2);
in = [a
      b];
out = zeros(size(in));
randNumber = floor(1.9*rand(1));
if (randNumber == 0)
    out = in;
elseif (randNumber == 1)
    a(1) = a2;
    a(2) = a1;
    b(1) = b2;
    b(2) = b1;
    out = [a
          b];
end

end

end

```

## **CHAPTER 6**

### **CONCLUSION AND FUTURE SCOPE**

#### **6.1 Conclusion**

The proposed system has been proven to be useful in providing bio metric authentication for access to confidential data and locations of the company. The system ensures that any data leak from the company side will not compromise the entire fingerprint of the employee as the database will only be storing one of the VCS shares. The process of LSB watermarking hides the transmission of shares in plain sight. We have learnt the process of encryption and decryption using VCS and steganography using LSB watermarking and have implemented the same in MATLAB.

#### **6.2 Future Scope**

We have currently implemented the system for only one type of biometric data,i.e. fingerprints. The system may be extended to incorporate other types of biometric data such as retina scans [4] which may require preprocessing in the form of feature extraction first.

The method of VCS implemented by us works on binary data. There exists other schemes of VCS which work on greyscale or even color images [7] and those may be used as and when appropriate.

Appendix



# CHAPTER 7

## APPENDIX

### 7.1 Matlab Code

#### 7.1.1 Encryption Using Modified Scheme

```
1  %Image Encryption using VCS and LSB Watermarking Random Algo
2  clc;
3  clear;
4  close all;
5
6  %Visual Cryptography
7  % Read Input Secret (fingerprint) Image
8
9  inImg = imread('102_3.tif');
10 figure;
11 imshow(inImg);title('Secret Image');
12
13 % Convert Secret Image to Binary
14
15 inImg = imbinarize(inImg);
16 figure;
17 imshow(inImg);title('Binary Secret Image');
18
19
20 % Perform Visual Cryptography
21 [share1, share2] = VisualCryptography(inImg);
22 disp("VCS Encryption")
23 figure;imshow(share1);title('Share 1');
24 figure;imshow(share2);title('Share 2');
25 % Save the shares
26 disp('Saving.....')
27 imwrite(share1,'Share1.tif');
```

```

28  imwrite(share2, 'Share2.tif');
29  disp('Done')
30
31  %LSB Watermarking
32  % LSB Watermarking of Share 2
33  disp('Performing LSB watermarking')
34  employee = 123456789;
35
36  % reshape to portrait
37  s = size(share2);
38  share2 = reshape(share2, [s(2), s(1)]);
39  share2 = imbinarize(share2);
40  figure;imshow(share2);title('Share 2 Rotated');
41
42  % Get Photo and resize it
43  face = imread('face.jpeg');
44  s = size(share2);
45  figure;imshow(face);title('Cover Image');
46  face = imresize(face, s);
47  figure;imshow(face);title('Resized Cover Image');
48
49  % Get the layer in which share will be hidden
50  x = rem(employee, 3)+1;
51  layer = face(:, :, x);
52  figure;subplot(121);imshow(layer);title('Cover Image (Image Layer)');
53      subplot(122);imshow(share2);title('Watermark (Share 2)');
54  % least significant bit substitution
55
56
57  % LSB1 or LSB2 randomly chosen
58
59  s = rng(employee)
60
61  a = randi([0 1], 600, 1);
62
63  share2_flip = zeros(600, 300);

```

```

64  for i = 1:size(share2,1)
65      if a(i) ==1
66          share2_flip(i,1:end) = fliplr(share2(i,1:end));
67      else
68          share2_flip(i,1:end) = share2(i,1:end);
69      end
70  end
71  share2_flip = imbinarize(share2_flip);
72  figure;imshow(share2_flip);title(['Fliped share 2']);
73
74
75  z=double(share2_flip);
76  r=double(layer-mod(layer,2)); % removal of LSB bits
77  l=uint8(r+z);
78
79  face(:, :,x) = l;
80  figure;imshow(face);title('Invisble watermarked Image');
81  disp('Lsb wartermarking done')
82
83
84
85  % Saving watermarked image
86  disp('Saving watermarked image')
87  imwrite(face,'face.tif');
88  disp('Done')
89
90
91  function [share1, share2] = VisualCryptography(inImg)
92  s = size(inImg);
93  share1 = zeros(s(1), (2 * s(2)));
94  share2 = zeros(s(1), (2 * s(2)));
95
96  % White Pixel Processing
97  disp('White Pixel Processing...');
98  % White Pixel share combinations
99

```

```

100  sla=[1 0];
101  slb=[1 0];
102  [x , y] = find(inImg == 1);
103  len = length(x);
104  for i=1:len
105      a=x(i);b=y(i);
106      pixShare=ShareGenerator(sla,slb);
107      share1((a),(2*b-1):(2*b))=pixShare(1,1:2);
108      share2((a),(2*b-1):(2*b))=pixShare(2,1:2);
109  end
110  % Black Pixel Processing
111  disp('Black Pixel Processing...');
112  % Black Pixel share combinations
113
114  s0a=[1 0];
115  s0b=[0 1];
116  [x ,y] = find(inImg == 0);
117  len = length(x);
118  for i=1:len
119      a=x(i);b=y(i);
120      pixShare=ShareGenerator(s0a,s0b);
121      share1((a),(2*b-1):(2*b))=pixShare(1,1:2);
122      share2((a),(2*b-1):(2*b))=pixShare(2,1:2);
123  end
124  disp('Share Generation Completed. ');
125  end
126
127  function out = ShareGenerator(a,b)
128  a1 = a(1);
129  a2 = a(2);
130  b1 = b(1);
131  b2 = b(2);
132  in = [a
133      b];
134  out = zeros(size(in));
135  randNumber = randi([0,1]);

```

```

136 if (randNumber == 0)
137     out = in;
138 elseif (randNumber == 1)
139     a(1) = a2;
140     a(2) = a1;
141     b(1) = b2;
142     b(2) = b1;
143     out = [a
144           b];
145 end
146
147 end
148
149

```

## 7.1.2 Decryption Using Modified Scheme

```

1  % Decryption
2  clc;
3  clear;
4  close all;
5
6  % Read the images
7  face = imread('face.tif');
8  share1 = imread('Share1.tif');
9  share1 = imbinarize(share1);
10
11 figure; subplot(121); imshow(face); title('Cover Image containing Share
    ↪ 2');
12     subplot(122); imshow(share1); title('Share 1');
13
14 employee = 123456789;
15 x = rem(employee,3)+1;
16 % inverse watermarking
17
18 layer = face(:, :, x);
19 a=mod(layer,2);

```

```

20 b=zeros(600,300);
21 for x=1:600
22     for y=1:300
23         if(a(x,y)==1)
24             b(x,y)=255;
25         end
26     end
27 end
28 recImg=imbinarize(b);
29 figure
30 imshow(recImg); % hidden image
31 title('Recovered hidden image (With flips)')
32
33
34 s = rng(employee)
35
36 a = randi([0 1],600,1);
37
38 share2_flip = zeros(600,300);
39 for i = 1:size(recImg,1)
40     if a(i) ==1
41         share2_flip(i,1:end) = fliplr(recImg(i,1:end));
42     else
43         share2_flip(i,1:end) = recImg(i,1:end);
44     end
45 end
46 share2 = imbinarize(share2_flip);
47 figure;imshow(share2_flip);title(['Fliped share 2']);
48
49 figure;imshow(share2);title('Share 2');
50
51 share2 = reshape(share2,[size(share2,2),size(share2,1)]);
52
53
54 figure;subplot(211);imshow(share1);title('Share 1');
55     subplot(212);imshow(share2);title('Share 2');

```

```

56
57
58  % Decrypting the shares
59  share12 = bitor(share1, share2);
60  share12 = ~share12;
61
62
63
64  s = [300 , 300];
65  shareop = zeros(s);
66  for i = 1:s(1)
67  for j = 1:s(2)
68      a = share12(i, (2*j-1):(2*j));
69      shareop(i, j) = max(a);
70  end
71  end
72  figure;imshow(share12);title('Overlapping Share 1 & 2');
73  disp("VCS Decryption")
74
75  figure;imshow(shareop);title('Decrypted output');
76
77  inImg = imread('102_3.tif');
78  inImg = imbinarize(inImg);
79  figure;imshow(inImg);title('Secret Image');
80
81  if shareop==inImg
82  disp('correct decoding')
83  end
84  imwrite(shareop, 'decrypted.tif');
85
86
87
88

```

## 7.2 Plagiarism Report

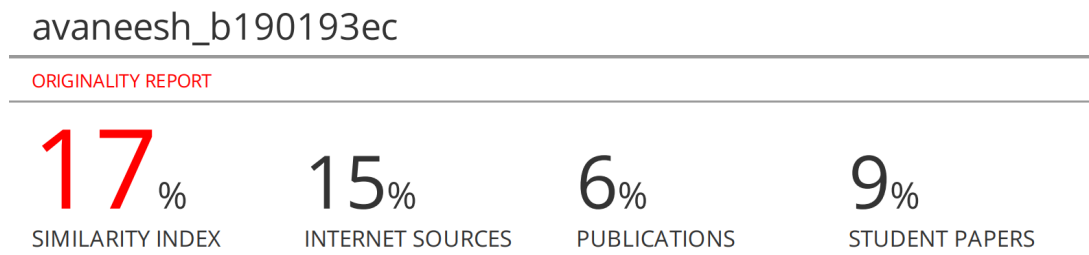


Figure 7.1: Plagiarism Report



## REFERENCES

- [1] M. Naor and A. Shamir, “Visual cryptography,” in *Workshop on the Theory and Application of Cryptographic Techniques*, pp. 1–12, Springer, 1994.
- [2] Y. Liang, H. V. Poor, S. Shamai, *et al.*, “Information theoretic security,” *Foundations and Trends® in Communications and Information Theory*, vol. 5, no. 4–5, pp. 355–580, 2009.
- [3] A. Ross and A. Othman, “Visual cryptography for biometric privacy,” *IEEE transactions on information forensics and security*, vol. 6, no. 1, pp. 70–81, 2010.
- [4] M. Suganya and K. Krishnakumari, “A novel retina based biometric privacy using visual cryptography,” *International Journal of Computer Science and Network Security (IJCSNS)*, vol. 16, no. 9, p. 76, 2016.
- [5] A. Mohanarathinam, S. Kamalraj, G. Prasanna Venkatesan, R. V. Ravi, and C. Manikandababu, “Digital watermarking techniques for image security: a review,” *Journal of Ambient Intelligence and Humanized Computing*, vol. 11, no. 8, pp. 3221–3229, 2020.
- [6] M. G. Almutiri and M. T. B. Othman, “Digital image watermarking based on lsb techniques: A comparative study,” *International Journal of Computer Applications*, vol. 975, p. 8887.
- [7] K. Anusree and G. Binu, “Biometric privacy using visual cryptography, halftoning and watermarking for multiple secrets,” in *2014 IEEE National Conference on Communication, Signal Processing and Networking (NCCSN)*, pp. 1–5, IEEE, 2014.