Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

DEPARTMENT OF INFORMATION TECHNOLOGY

COURSE CODE: DJS22ITL504 **DATE: 4/08/24**
COURSE NAME: Cryptography and Network Security Laboratory **CLASS: TYBTech**
NAME: Anish Sharma

EXPERIMENT NO. 1

CO/LO: Design secure system using appropriate security mechanism

AIM / OBJECTIVE:

a. Implementation of Ceaser Cipher on alphanumeric data.
b. Implementation of Ceaser Cipher on gray scale image.

.

THEORY / CONCEPT / ALGORITHM:

The Caesar Cipher is a basic encryption technique in which each letter in the plaintext is shifted a fixed number of positions down or up the alphabet. The shift value, known as the key, is the same for both encryption and decryption. For example, with a shift of 3, 'A' becomes 'D', 'B' becomes 'E', and so on. If the end of the alphabet is reached, the cipher wraps around to the beginning

The Caesar Cipher can be adapted for grayscale image encryption by treating each pixel value as a character in the algorithm. In a grayscale image, each pixel intensity ranges from 0 to 255. Encryption involves shifting each pixel's intensity by a fixed key value, wrapping around using modulo 256 arithmetic. Mathematically, for a pixel value ppp, encryption is E(p)=(p+k)mod 256E(p) = (p + k) \mod 256E(p)=(p+k)mod256 and decryption is D(p)=(p−k)mod 256D(p) = (p - k) \mod 256D(p)=(p−k)mod256, where kkk is the key. This technique provides basic encryption, making the image appear noisy and obscured, though it is not highly secure.

SOURCE CODE:

a) Implementation of Ceaser Cipher on alphanumeric data.

```
def encrypt_decrypt_string(input_string, key):
    key = int(key) % 256
    encrypted_decrypted_chars = [chr(ord(char) + key) for char in
input_string]
    return ''.join(encrypted_decrypted_chars)

def decrypt_string(input_string, key):
    key = int(key) % 256
    encrypted_decrypted_chars = [chr(ord(char) - key) for char in
input_string]
    return ''.join(encrypted_decrypted_chars)


def main():
    input_string = input("Enter the string to be encrypted/decrypted: ")
    key = input("Enter a numeric key for encryption and decryption: ")

    encrypted_string = encrypt_decrypt_string(input_string, key)
    print(f"Encrypted string: {encrypted_string}")

    decrypted_string = decrypt_string(encrypted_string, key)
    print(f"Decrypted string: {decrypted_string}")

if __name__ == "__main__":
    main()
```

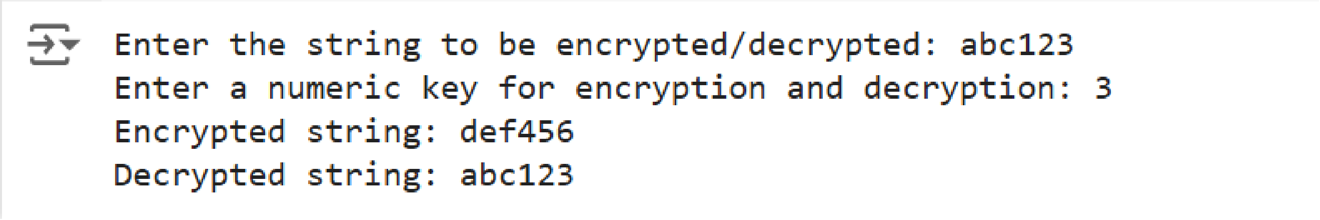
b) Implementation of Ceaser Cipher on gray scale image

```
from PIL import Image
import numpy as np

def load_image(image_path):
    image = Image.open('grayscale.jpeg').convert('L')
    return np.array(image)


def save_image(image_array, output_path):
    image = Image.fromarray(image_array.astype(np.uint8))
    image.save(output_path)

def encryp  t_decrypt_image(image_array, key):
    key = int(key) % 256
    encrypted_image_array = np.bitwise_xor(image_array, key)
    return encrypted_image_array
```

```
def compute_difference_image(original_image, decrypted_image):
    difference_image = np.abs(original_image - decrypted_image)
    return difference_image

def main():
    input_image_path = 'grayscale.jpeg'
    output_encrypted_image_path = 'encrypted.jpeg'
    output_decrypted_image_path = 'decrypted.jpeg'
    difference_image_path = 'difference.jpeg'
    key = input("Enter a numeric key for encryption and decryption: ")

    image_array = load_image(input_image_path)

    encrypted_image_array = encrypt_decrypt_image(image_array, key)
    save_image(encrypted_image_array, output_encrypted_image_path)
    print(f"Encrypted image saved as {output_encrypted_image_path}")

    decrypted_image_array = encrypt_decrypt_image(encrypted_image_array, key)
    save_image(decrypted_image_array, output_decrypted_image_path)
    print(f"Decrypted image saved as {output_decrypted_image_path}")

    difference_image_array = compute_difference_image(image_array,
decrypted_image_array)
    save_image(difference_image_array, difference_image_path)
    print(f"Difference image saved as {difference_image_path}")

if __name__ == "__main__":
    main()
```

SAMPLE INPUT AND OUTPUT:

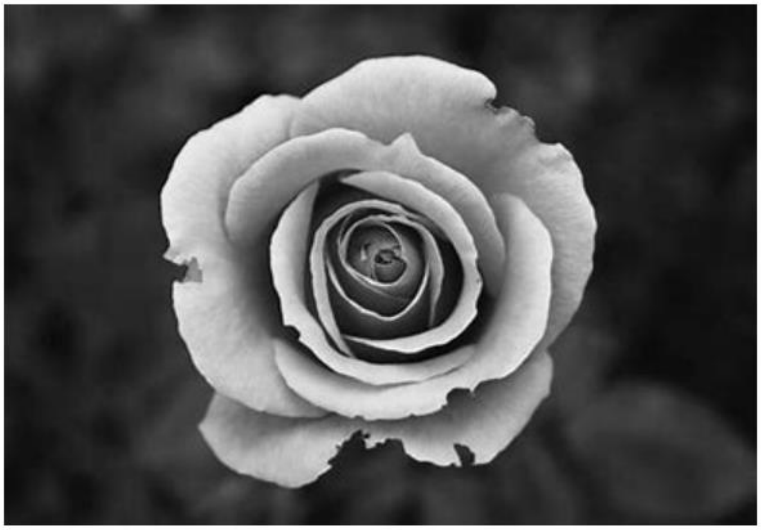
a) Implementation of Ceaser Cipher on alphanumeric data.



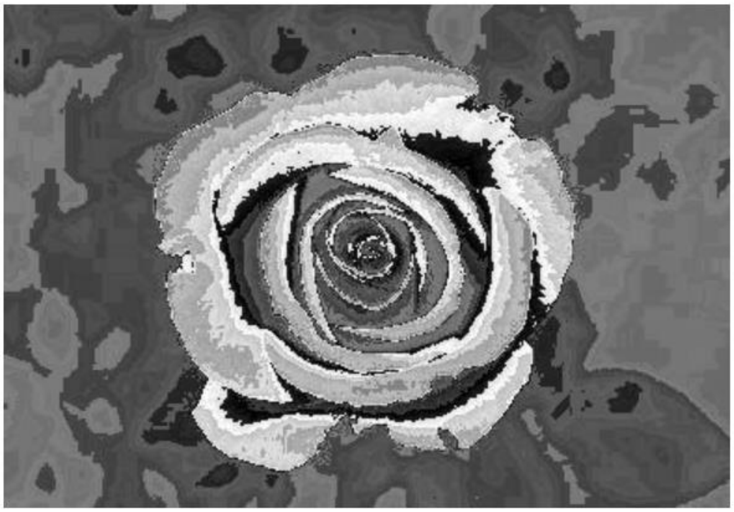
```
Enter the string to be encrypted/decrypted: abc123
Enter a numeric key for encryption and decryption: 3
Encrypted string: def456
Decrypted string: abc123
```

b) Implementation of Ceaser Cipher on gray scale image

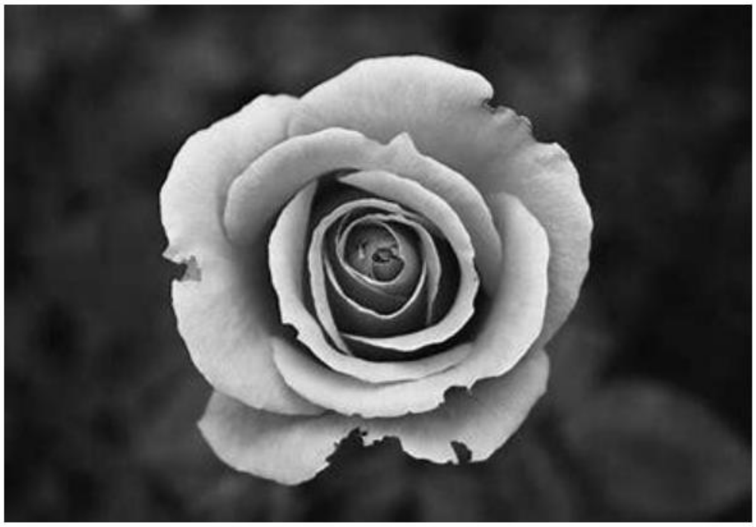
Original Image



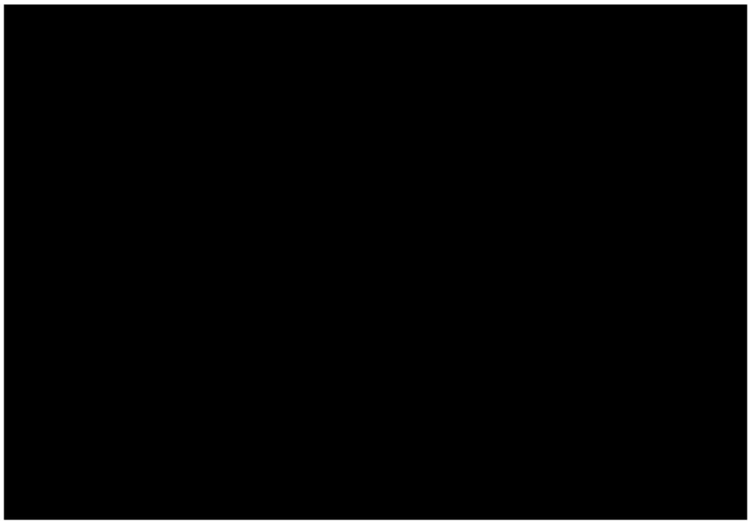
Encrypted image

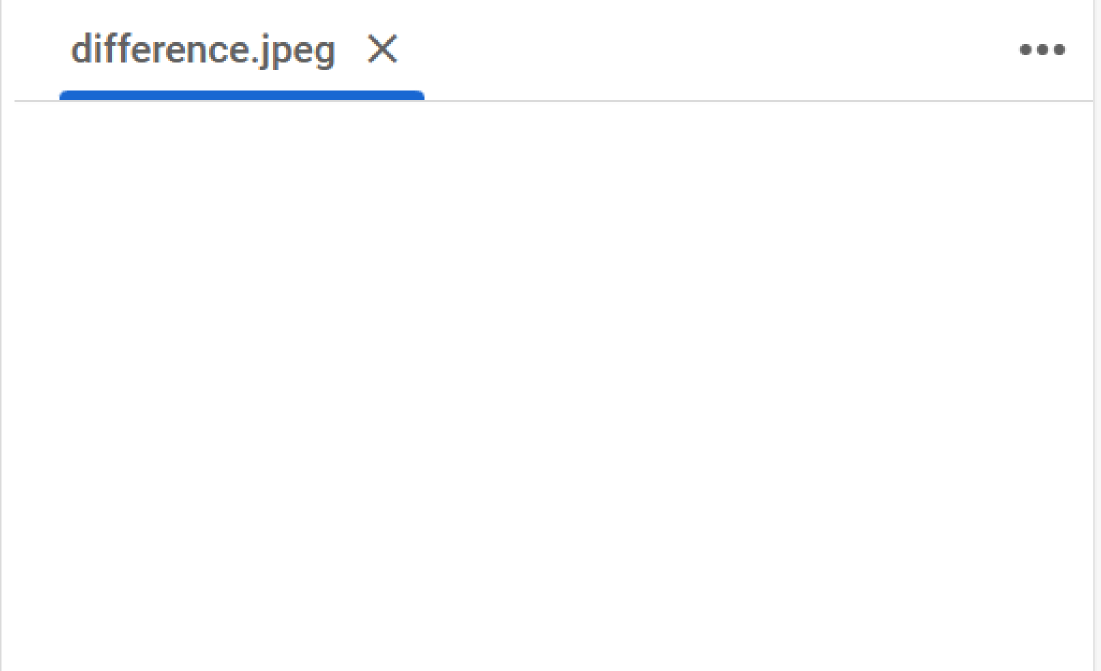


Decrypted image



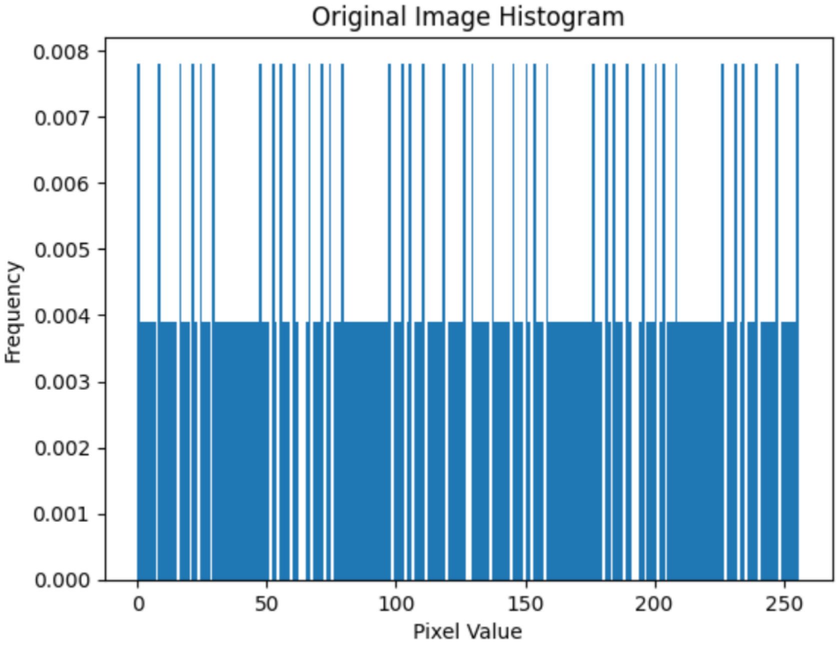
Difference images

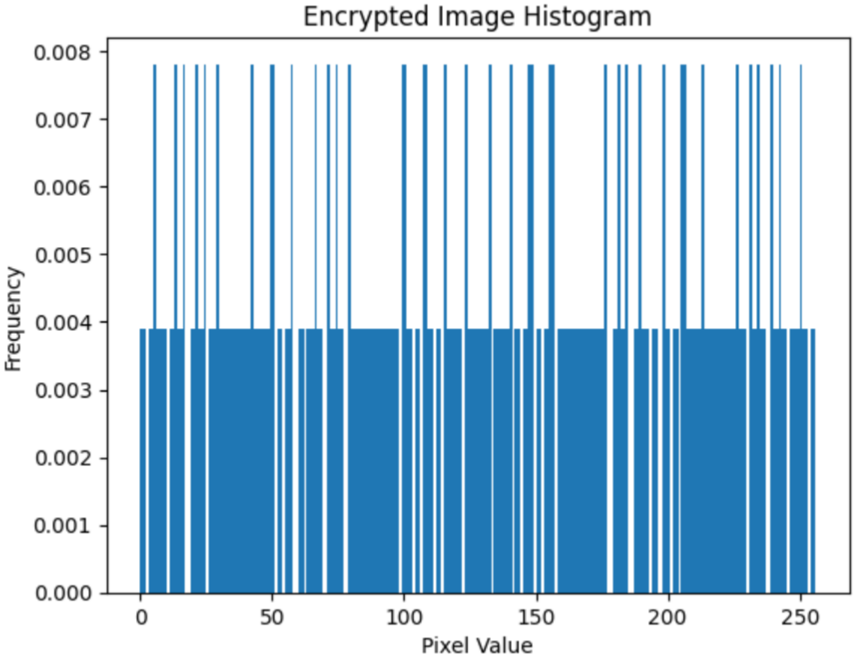


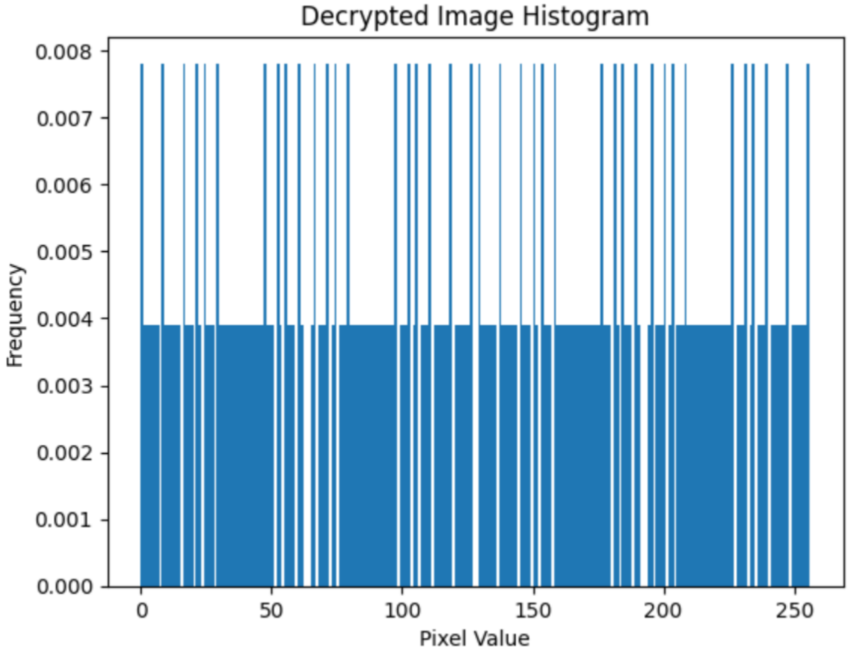


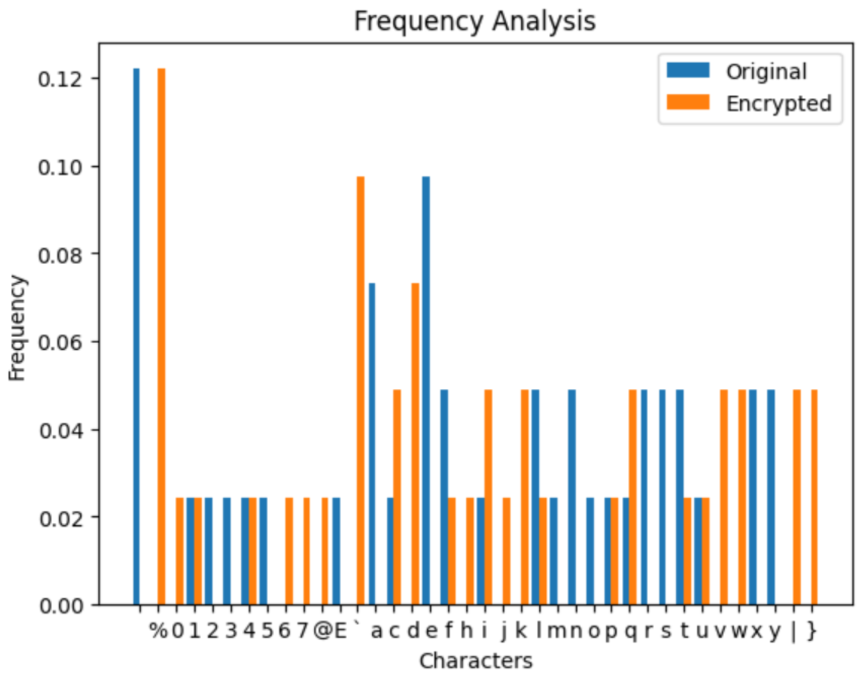
QUESTIONS:

1. Perform a frequency analysis of the encrypted alphanumeric data and compare it to the frequency of the original data.
2. Generate histograms of the pixel values for the original, encrypted, and decrypted images.




Encrypted Image Histogram


Decrypted Image Histogram



CONCLUSION: We performed Ceaser Cipher encryption and decryption algorithms on text and grayscale images

REFERENCES:

Pillow (PIL):

- Official documentation: Pillow Documentation