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)

Experiment 6

Date of Performance : 20 March 2023

Date of Submission: 20 March 2023

SAP Id: 60004200132

Name : Ayush Jain

Div: B

Batch : B3

Aim of Experiment

Implement RSA cryptosystem. Demonstrate the application of RSA cryptosystem using multimedia data.

Theory / Algorithm / Conceptual Description:

RSA (Rivest-Shamir-Adleman) is a public-key cryptosystem that is widely used for secure communication over the internet. The basic idea behind RSA is based on the mathematical properties of large prime numbers and their factorization.

RSA is based on the following three algorithms:

1. Key generation: In this algorithm, two large prime numbers (p and q) are selected and their product (n=p*q) is calculated. Then, a number e is chosen such that it is relatively prime to (p-1)*(q-1). The public key is (n,e), and the private key is (p,q,d) where d is the modular multiplicative inverse of e modulo (p-1)*(q-1).
2. Encryption: In this algorithm, the message is represented as a number (m) and is encrypted using the public key (n,e) as follows: c = m^e mod n, where c is the encrypted message.
3. Decryption: In this algorithm, the encrypted message (c) is decrypted using the private key (p,q,d) as follows: m = c^d mod n, where m is the original message.

The security of RSA is based on the difficulty of factoring the product of two large prime numbers. If an attacker can factor n into its prime factors, then they can calculate d and decrypt the message. However, the best-known algorithms for factoring large numbers have exponential time complexity, making it infeasible to break RSA encryption for large enough key sizes.

In summary, RSA is a widely used public-key cryptosystem based on the mathematical properties of large prime numbers and their factorization. Its security relies on the difficulty of factoring large numbers, which is infeasible using current known algorithms for large enough key sizes.

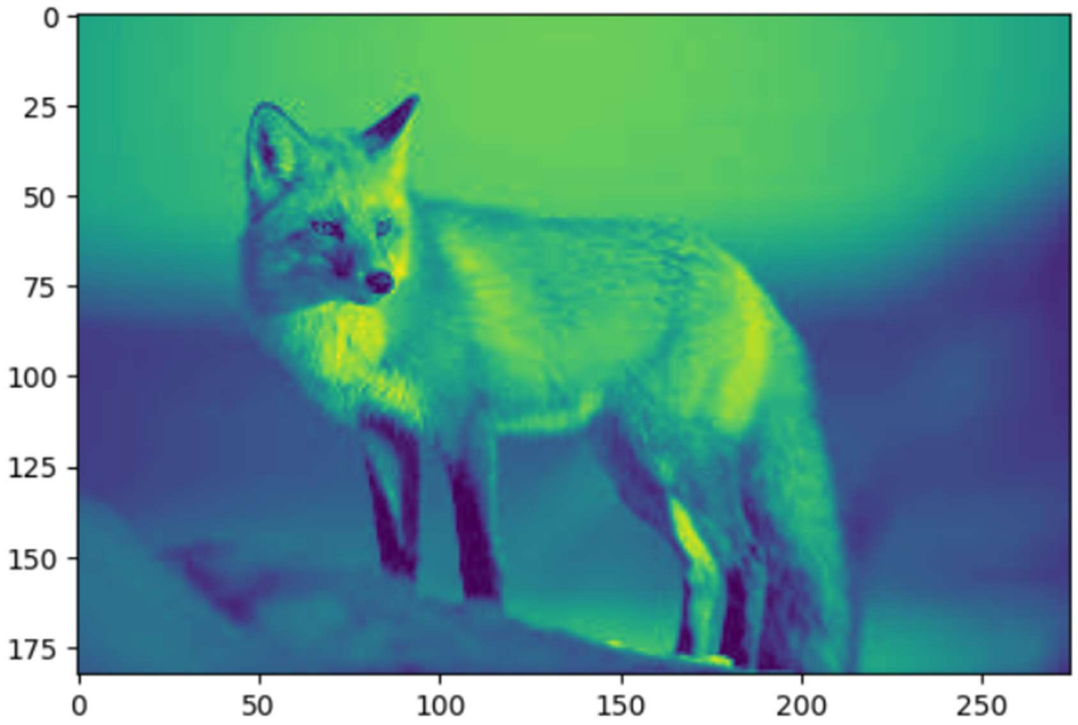
Program

```
import numpy as np
import random
import cv2
import matplotlib.pyplot as plt
# Select 2 prime numbers
p = 19
q = 13
# First public key
n = p * q
n
def gcd(a, b):

    if (a == 0):
        return b
    return gcd(b % a, a)
def phi(n):

    result = 1
    for i in range(2, n):
        if (gcd(i, n) == 1):
            result+=1
    return result
# Calculating Phi(n)
phi_n = phi(n)
phi_n
# e = random.randint(0, phi_n)
e = 5
pulbic_key  = (n, e)
# k = random.randint(1, 10)
k = 4
d = (k * phi_n + 1) / (e)
d
private_key = d
image = cv2.imread('/content/animal.jpeg', 0)
image.shape
plt.imshow(image)

# display that image
plt.show()
def encrypt(input, e = 5, n = 247):
```

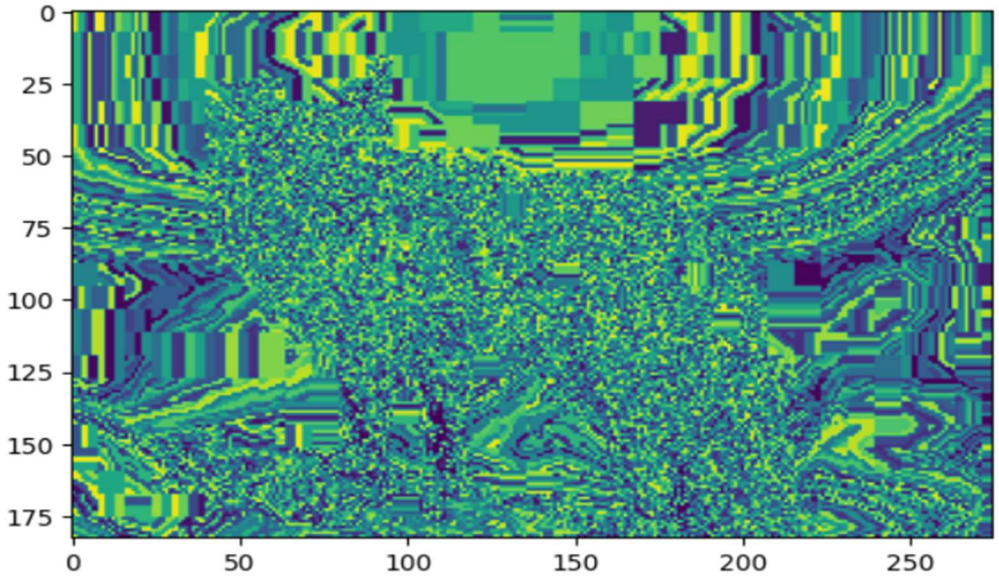
```
    cipher = pow(input, e) % n
    return cipher
shape = image.shape
image
pixels = image.flatten()
enc = []

for pixel in pixels:
    enc.append(encrypt(int(pixel)))

enc = np.array(enc)

encrypted_image = enc.reshape(shape)
encrypted_image
cv2.imwrite('rsa_encryption.png', encrypted_image)
plt.imshow(encrypted_image)

# display that image
plt.show()
def decryption(encrypted, d=173, n=247):
    return pow(encrypted, int(d)) % n
image2 = cv2.imread('rsa_encryption.jpg', 0)
pixels_enc = encrypted_image.flatten()
image2
og = []

for pixel in pixels_enc:
    og.append(decryption(int(pixel)))

og = np.array(og)

original_image = og.reshape(shape)
original_image
cv2.imwrite('decrypted.png', original_image)
plt.imshow(original_image)
# display that image
plt.show()
```

Output

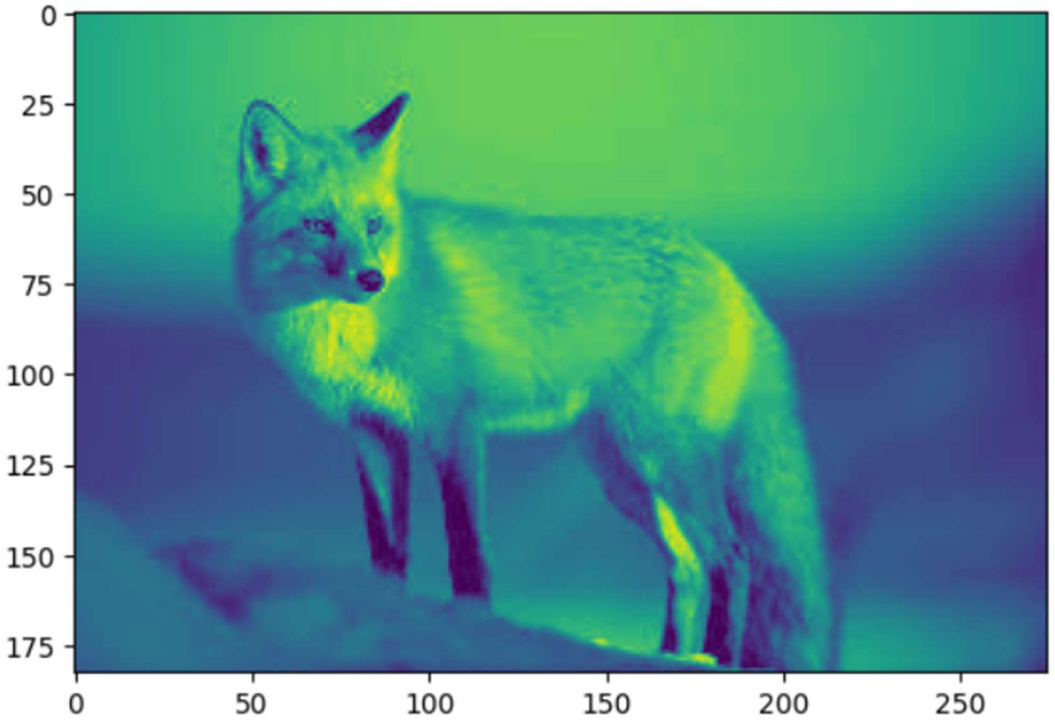
Input



Encrypted Image



Decrypted Image



CONCLUSION: We have successfully implemented RSA on multimedia.