**Experiment 5**

**Date of Performance :**  **Date of Submission:**

**SAP Id: 60004190057** **Name : Junaid Altaf Girkar**

**Div:** **A** **Batch : A4**

**Aim of Experiment**

Design and Implement Encryption and Decryption Algorithm for RSA Algorithm

**Theory / Algorithm / Conceptual Description**

RSA algorithm is an asymmetric cryptography algorithm. Asymmetric actually means that it works on two different keys i.e. Public Key and Private Key. As the name describes, the Public Key is given to everyone and the Private key is kept private.

The idea of RSA is based on the fact that it is difficult to factorize a large integer. The public key consists of two numbers where one number is multiplication of two large prime numbers. And private keys are also derived from the same two prime numbers. So if somebody can factorize the large number, the private key is compromised. Therefore encryption strength totally lies on the key size and if we double or triple the key size, the strength of encryption increases exponentially. RSA keys can be typically 1024 or 2048 bits long, but experts believe that 1024 bit keys could be broken in the near future. But till now it seems to be an infeasible task.

**PUBLIC KEY GENERATION:**

| Select two prime no's. Suppose P = 53 and Q = 59. Now First part of the Public key : n = P\*Q = 3127.  We also need a small exponent say e :   But e Must be An integer.  Not be a factor of n.   1 < e < Φ(n) [Φ(n) is discussed below],  Let us now consider it to be equal to 3. Our Public Key is made of n and e |
| --- |
|  |

**PRIVATE KEY GENERATION**

| We need to calculate Φ(n) : Such that Φ(n) = (P-1)(Q-1)   so, Φ(n) = 3016   Now calculate Private Key, d :  d = (k\*Φ(n) + 1) / e for some integer k For k = 2, value of d is 2011. |
| --- |

**ENCRYPTION**:

| Convert letters to numbers : H = 8 and I = 9   Thus Encrypted Data c = 89e mod n.  Thus our Encrypted Data comes out to be 1394 |
| --- |

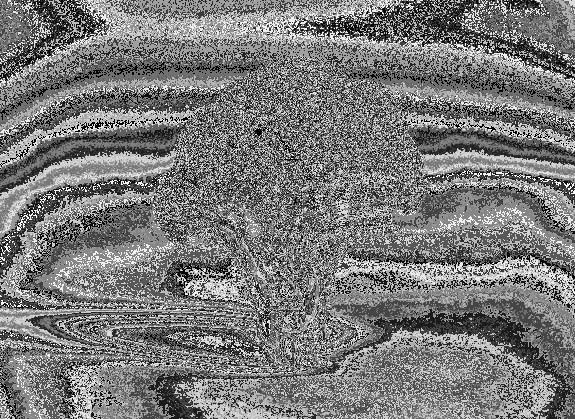
**DECRYPTION**:

| **Now we will decrypt 1394 :  Decrypted Data = cd mod n.  Thus our Encrypted Data comes out to be 89 8 = H and I = 9 i.e. "HI".** |
| --- |

**CODE**:

| 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  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) # 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) private\_key = d  image = cv2.imread('RSA.jpg', 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 pixels = image.flatten() enc = []  for pixel in pixels:  enc.append(encrypt(int(pixel)))  enc = np.array(enc)  encrypted\_image = enc.reshape(shape)  cv2.imwrite('rsa\_encryption.jpg', encrypted\_image) def decryption(encrypted, d=173, n=247):  return pow(encrypted, int(d)) % n  #DECRYPTION: 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.jpg', original\_image) plt.imshow(original\_image, cmap='gray') # display that image plt.show() |
| --- |

**OUTPUT**:



ORIGINAL ENCRYPTED DECRYPTED

**CONCLUSION**

With the increasing amount of data being generated, it is very important that confidential information does not get leaked and is read by the intended recipient.We learnt about asymmetric key ciphers and the RSA algorithm and we then wrote a python program to implement it.