Department of Computer Science and Engineering

Experiment Name: RSA ALGORITHM

Experiment No: 9

Date of perform: Jan 17, 2023
Date of submission: May 13, 2024

Submitted to:

Md. Imdadul Islam
Professor of CSE, Jahangirnagar University
Mohammad Ashraful Islam
Assistant Professor of CSE,Jahangirnagar University

Submitted by:

Name: Sudipta Singha
Exam Roll: 202220
Class Roll: 408

Jahangirnagar University,Savar, Dhaka

1 Objective

The goal of the lab is to test RSA algorithm for encrypting and decrypting an Image.For this we use python language.

2 Procedure

```
1
2  import math
3  import random
4  from PIL import Image
5
6  def generate_primes(n):
7      primes = []
8      for num in range(2, n):
9          prime = True
10         for i in range(2, int(math.sqrt(num)) + 1):
11             if (num % i) == 0:
12                 prime = False
13                 break
14         if prime:
15             primes.append(num)
16     return primes
17
18 def generate_keys(p, q):
19     n = p * q
20     phi = (p - 1) * (q - 1)
21
22     e = random.randrange(1, phi)
23     g = math.gcd(e, phi)
24     while g != 1:
25         e = random.randrange(1, phi)
26         g = math.gcd(e, phi)
27
28     d = mod_inverse(e, phi)
29
30     return ((e, n), (d, n))
31
32 def mod_inverse(a, m):
33     m0, x0, x1 = m, 0, 1
34     while a > 1:
35         q = a // m
36         m, a = a % m, m
37         x0, x1 = x1 - q * x0, x0
38     return x1 + m0 if x1 < 0 else x1
39
40 def encrypt_image(image_path, public_key):
41     image = Image.open(image_path)
42     width, height = image.size
43     pixels = list(image.getdata())
44
45     e, n = public_key
46     encrypted_pixels = []
47     for pixel in pixels:
48         encrypted_pixel = tuple(pow(component, e, n) for component in pixel)
49         encrypted_pixels.append(encrypted_pixel)
50
```

```
    return encrypted_pixels, width, height

def save_encrypted_image(encrypted_pixels, width, height, output_path):
    encrypted_image = Image.new('RGB', (width, height))
    encrypted_image.putdata(encrypted_pixels)
    encrypted_image.save(output_path)

def decrypt_image(encrypted_pixels, private_key):
    d, n = private_key
    decrypted_pixels = []
    for pixel in encrypted_pixels:
        decrypted_pixel = tuple(pow(component, d, n) for component in pixel)
        decrypted_pixels.append(decrypted_pixel)

    return decrypted_pixels

def save_decrypted_image(decrypted_pixels, width, height, output_path):
    decrypted_image = Image.new('RGB', (width, height))
    decrypted_image.putdata(decrypted_pixels)
    decrypted_image.save(output_path)

def main():
    image_path = 'image.jpg'
    output_path_encrypted = 'encrypted_image.png'
    output_path_decrypted = 'decrypted.jpg'

    primes = generate_primes(100)
    p, q = random.choice(primes), random.choice(primes)

    public_key, private_key = generate_keys(p, q)

    encrypted_pixels, width, height = encrypt_image(image_path, public_key)

    save_encrypted_image(encrypted_pixels, width, height, output_path_encrypted)

    decrypted_pixels = decrypt_image(encrypted_pixels, private_key)

    save_decrypted_image(decrypted_pixels, width, height, output_path_decrypted)

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

3 Result

Figure 1: Original image jpg

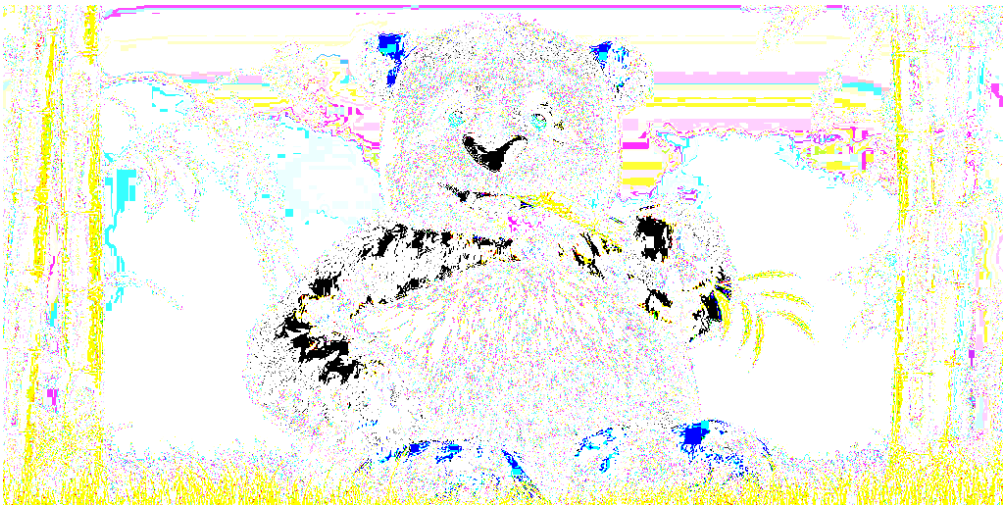

Figure 2: encrypted image png

Figure 3: decrypted jpg