**RSA Encryption and Decryption Program using Python**

Python Code for Stage 1 - RSA Key Generation:

import random

def gcd(a, b):

while b != 0:

a, b = b, a % b

return a

def extended\_gcd(a, b):

if a == 0:

return (b, 0, 1)

else:

g, y, x = extended\_gcd(b % a, a)

return (g, x - (b // a) \* y, y)

def mod\_inverse(a, m):

g, x, y = extended\_gcd(a, m)

if g != 1:

raise ValueError("Modular inverse does not exist.")

else:

return x % m

def generate\_key\_pair(bit\_length):

p = q = 1

while bit\_length > p.bit\_length() + q.bit\_length():

p = random.getrandbits(bit\_length // 2)

q = random.getrandbits(bit\_length - p.bit\_length())

while not (prime\_check(p) and prime\_check(q)):

p = random.getrandbits(bit\_length // 2)

q = random.getrandbits(bit\_length - p.bit\_length())

n = p \* q

phi\_n = (p - 1) \* (q - 1)

e = random.randrange(2, phi\_n)

while gcd(e, phi\_n) != 1:

e = random.randrange(2, phi\_n)

d = mod\_inverse(e, phi\_n)

public\_key = (n, e)

private\_key = (n, d)

return public\_key, private\_key

def prime\_check(num):

if num < 2:

return False

for i in range(2, int(num\*\*0.5) + 1):

if num % i == 0:

return False

return True

if \_\_name\_\_ == "\_\_main\_\_":

key\_length = 16

public\_key, private\_key = generate\_key\_pair(key\_length)

print("Public Key (n, e):", public\_key)

print("Private Key (n, d):", private\_key)

Python Code for Stage 2 - Encryption/Decryption:

def encrypt(message, public\_key):

n, e = public\_key

encrypted\_message = [pow(ord(char), e, n) for char in message]

return encrypted\_message

def decrypt(encrypted\_message, private\_key):

n, d = private\_key

decrypted\_message = ''.join([chr(pow(char, d, n)) for char in encrypted\_message])

return decrypted\_message

if \_\_name\_\_ == "\_\_main\_\_":

# Assuming plaintext.txt contains the text you want to encrypt

with open("plaintext.txt", "r") as file:

plaintext = file.read()

public\_key = (YOUR\_N\_VALUE, YOUR\_E\_VALUE) # Replace with the actual public key values obtained from key generation

# Encryption

encrypted\_text = encrypt(plaintext, public\_key)

with open("ciphertext.txt", "w") as file:

file.write(' '.join(map(str, encrypted\_text)))

private\_key = (YOUR\_N\_VALUE, YOUR\_D\_VALUE) # Replace with the actual private key values obtained from key generation

# Decryption

with open("ciphertext.txt", "r") as file:

ciphertext = list(map(int, file.read().split()))

decrypted\_text = decrypt(ciphertext, private\_key)

with open("decoded.txt", "w") as file:

file.write(decrypted\_text)