Simple RSA Encryption and Decryption in Python

https://github.com/Aniket-Choudhari-01/rsa-with-pyodide

REPORT

Choudhari Aniket Baban
CS22B1010

Implementation

RSA is an asymmetric cryptographic algorithm that uses two keys — public for encryption and private for decryption — based on the difficulty of factoring large primes.

The key steps for the implementation include:

- Generate two large prime numbers p and q.
- Compute $n = p \times q$ and $\phi(n) = (p-1)*(q-1)$.
- Select e such that $\gcd(e, \phi(n)) = 1$.
- Calculate private key d as modular inverse of e mod $\phi(n)$.
- Encrypt: cipher = $(\text{message})^e$ mod n.
- Decrypt: message = $(\text{cipher})^d$ mod n.

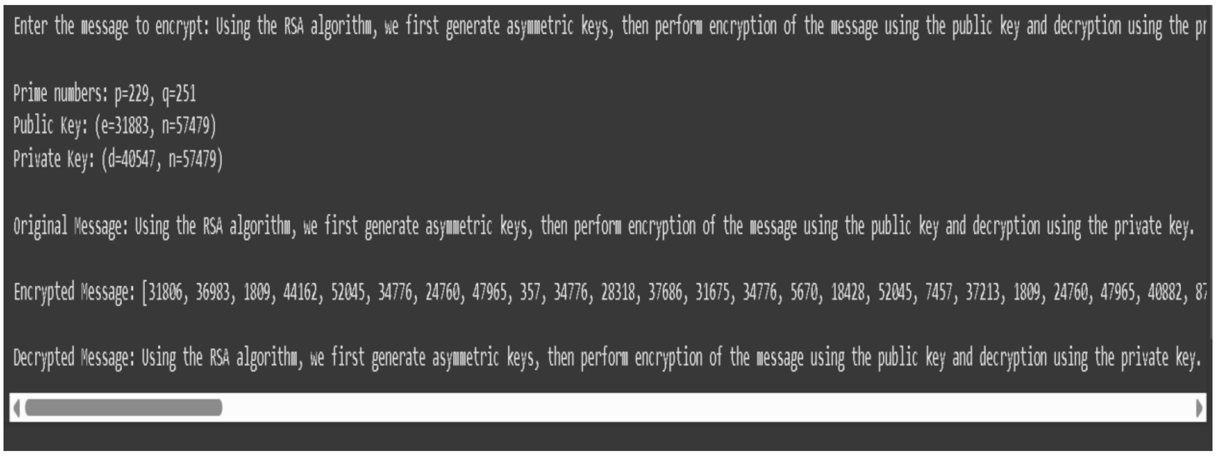
Code

```
import random
from sympy import isprime, mod_inverse

def generate_prime(bits=8):
    while True:
        num = random.randint(2**(bits-1), 2**bits - 1)
        if isprime(num):
            return num

def rsa_key_generation():
    p = generate_prime()
    q = generate_prime()
    n = p * q
    phi_n = (p - 1) * (q - 1)

    e = random.randint(2, phi_n - 1)
    while gcd(e, phi_n) != 1:
        e = random.randint(2, phi_n - 1)

    d = mod_inverse(e, phi_n)
```

```python
    return (p, q, n, phi_n, e, d)

def encrypt(message, e, n):
    numerical_message = [ord(char) for char in message]
    cipher_text = [pow(num, e, n) for num in numerical_message]
    return cipher_text

def decrypt(cipher_text, d, n):
    decrypted_numerical_message = [pow(num, d, n) for num in cipher_text]
    decrypted_message = ''.join([chr(num) for num in decrypted_numerical_message])
    return decrypted_message

def gcd(a, b):
    while b:
        a, b = b, a % b
    return a

if __name__ == "__main__":
    p, q, n, phi_n, e, d = rsa_key_generation()

    message = input("Enter the message to encrypt: ")

    cipher_text = encrypt(message, e, n)
    decrypted_message = decrypt(cipher_text, d, n)

    print(f"\nPrime numbers: p={p}, q={q}")
    print(f"Public Key: (e={e}, n={n})")
    print(f"Private Key: (d={d}, n={n})")
    print(f"\nOriginal Message: {message}")
    print(f"\nEncrypted Message: {cipher_text}")
    print(f"\nDecrypted Message: {decrypted_message}")
```

Results

The implemented RSA encryption and decryption algorithm was successfully tested with various input messages. The code accurately generates asymmetric keys, encrypts the input, and correctly retrieves the original message after decryption, demonstrating the correctness and reliability of the RSA process.



```
Enter the message to encrypt: Using the RSA algorithm, we first generate asymmetric keys, then perform encryption of the message using the public key and decryption using the pr

Prime numbers: p=229, q=251
Public Key: (e=31883, n=57479)
Private Key: (d=40547, n=57479)

Original Message: Using the RSA algorithm, we first generate asymmetric keys, then perform encryption of the message using the public key and decryption using the private key.

Encrypted Message: [31806, 36983, 1809, 44162, 52045, 34776, 24760, 47965, 357, 34776, 28318, 37686, 31675, 34776, 5670, 18428, 52045, 7457, 37213, 1809, 24760, 47965, 40882, 87

Decrypted Message: Using the RSA algorithm, we first generate asymmetric keys, then perform encryption of the message using the public key and decryption using the private key.
```