Experiment No. 4

Title: RSA Cipher

Batch: B2 **Roll No.: 16010421119** **Experiment No.: 4**

Aim: To implement RSA cipher

Resources needed: Windows/Linux

Theory: Pre Lab/ Prior Concepts:

RSA is a public key algorithm named after its inventers Rivest, Shamir and Adleman. The characteristics of public key cryptography (which is also called as Asymmetric Cryptography) are as below:

- It has two keys. One key is called as private key and the other one is called as public key. Everyone who uses this cryptography has to have two keys each.
- Keys used for encryption and decryption should be different. If one is used for encryption, then other must be used for decryption. Any key can be used for encryption and then remaining key can be used for decryption.
- Public Key Cryptography is based on solid foundation of mathematics.
- It has large computational overheads. Hence ciphertext generated is of much larger size than the plaintext. Hence it is normally used for encrypting small size of data blocks. For example, passwords, symmetric keys, etc. It is not preferred to encrypt large files.
- RSA algorithm gets its security from the fact that it is extremely difficult to factorize large prime number.
- Security of the algorithm depends on the size of the key. Greater the size of the key, larger is the security. The key length is variable. The most commonly used key length is 512 bits.

Procedure / Approach /Algorithm / Activity Diagram:

A. Key generation Algorithm:

1. Choose large prime numbers p and q.
2. Calculate product n= pq. The value of n can be revealed publicly, but n is large enough that even supercomputers cannot factor it in a reasonable amount of time (years or even centuries).
3. Calculate phi = (p-1)(q-1)
4. Select e < phi such that it is relatively prime to phi. The public key is (e, n).
5. Determine d such that ed = 1 mod phi. The private key is (d, n). d, p ,q and phi are kept secret, only (e,n) is made public.

B. Encryption:

6. Suppose M is the message.
7. Encrypt this message using public key of the receiver as follows
   C = M^e mod n.

C. Decryption:

8. Decrypt the C generated in step 7 using private key of the receiver as follows M = C^d mod n.

Results: (Program printout as per the format)

CODE:

1. Encryption

```python
def encrypt(public_key, message):
    n, e = public_key
    encrypted = []

    for char in message:
        char_value = ord(char)
        encrypted_value = pow(char_value, e, n)
        encrypted.append(encrypted_value)

    return encrypted

def print_encrypted(encrypted):
    print("Encrypted message:", end=' ')
    for value in encrypted:
        print(value, end=' ')
    print()
```

2. Decryption

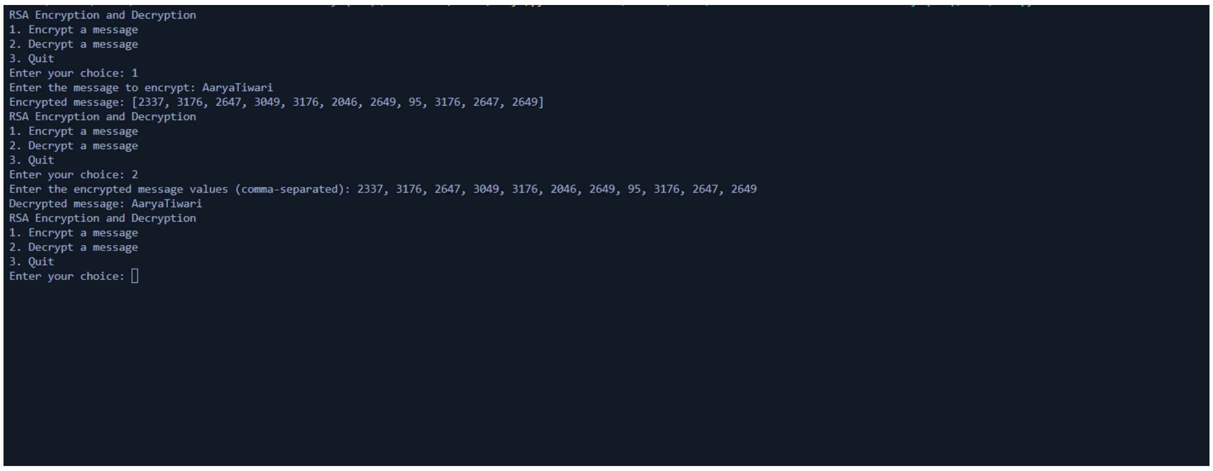
```python
def decrypt(private_key, encrypted):
    n, d = private_key
    decrypted = []

    for encrypted_value in encrypted:
        char_value = pow(encrypted_value, d, n)
        decrypted_char = chr(char_value)
        decrypted.append(decrypted_char)
    return ''.join(decrypted)
```

3. Menu-Driven Program

```
import random

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

def multiplicative_inverse(e, phi):
    d = 0
    x1, x2 = 0, 1
    y1, y2 = 1, 0
    while phi:
        quotient = e // phi
        e, phi = phi, e % phi
        x1, x2 = x2 - quotient * x1, x1
        y1, y2 = y2 - quotient * y1, y1
    d = x2
    return d

def generate_keypair(p, q):
    n = p * q
    phi = (p - 1) * (q - 1)
    e = random.randrange(2, phi)
    while gcd(e, phi) != 1:
        e = random.randrange(2, phi)
    d = multiplicative_inverse(e, phi)

    return ((n, e), (n, d))

def encrypt(public_key, message):
    n, e = public_key
    encrypted = []

    for char in message:
        char_value = ord(char)
        encrypted_value = pow(char_value, e, n)
        encrypted.append(encrypted_value)

    return encrypted

def print_encrypted(encrypted):
    print("Encrypted message:", end=' ')
    for value in encrypted:
        print(value, end=' ')
    print()

def decrypt(private_key, encrypted):
    n, d = private_key
    decrypted = []

    for encrypted_value in encrypted:
```

```python
        char_value = pow(encrypted_value, d, n)
        decrypted_char = chr(char_value)
        decrypted.append(decrypted_char)
    return ''.join(decrypted)

def main_menu():
    print("RSA Encryption and Decryption")
    print("1. Encrypt a message")
    print("2. Decrypt a message")
    print("3. Quit")
    choice = int(input("Enter your choice: "))
    return choice

if __name__ == '__main__':
    p = 61
    q = 53
    public_key, private_key = generate_keypair(p, q)

    while True:
        choice = main_menu()

        if choice == 1:
            message = input("Enter the message to encrypt: ")
            encrypted_message = encrypt(public_key, message)
            # print_encrypted(encrypted_message)
            print("Encrypted message:", encrypted_message)
        elif choice == 2:
            encrypted_message = list(map(int, input("Enter the encrypted 
message values (comma-separated): ").split(',')))
            decrypted_message = decrypt(private_key, encrypted_message)
            print("Decrypted message:", decrypted_message)
        elif choice == 3:
            print("Exiting the program.")
            break
        else:
            print("Invalid choice. Please select again.")
```

OutPut:



```
RSA Encryption and Decryption
1. Encrypt a message
2. Decrypt a message
3. Quit
Enter your choice: 1
Enter the message to encrypt: AaryaTiwari
Encrypted message: [2337, 3176, 2647, 3049, 3176, 2046, 2649, 95, 3176, 2647, 2649]
RSA Encryption and Decryption
1. Encrypt a message
2. Decrypt a message
3. Quit
Enter your choice: 2
Enter the encrypted message values (comma-separated): 2337, 3176, 2647, 3049, 3176, 2046, 2649, 95, 3176, 2647, 2649
Decrypted message: AaryaTiwari
RSA Encryption and Decryption
1. Encrypt a message
2. Decrypt a message
3. Quit
Enter your choice: 
```

Questions:

1. In RSA cryptosystem each plaintext character is presented by the number between 00(A) and 25(Z). The number 26 represents the blank character. Bob wants to send Alice the message “Hello World”. So the plaintext is as below,
   07 04 11 11 14 26 22 14 17 11 03 . Suppose p=11, q=3. Generate receiver’s key pair and show encryption and decryption of the message using RSA cipher.
   Ans:
   Public Key: (33, 17)
   Private Key: (33, 17)

   Using RSA Cipher the encrypted message is:
   [16, 30, 13, 13, 22, 26, 11, 22, 28, 13, 12]

2. List the attacks on RSA.

Ans:

1. **Brute Force Attack:** An attacker tries all possible private keys until the correct one is found. This attack is practically infeasible if the key size is sufficiently large.

2. **Factorization Attack:** This attack exploits the difficulty of factoring the product of two large prime numbers. If an attacker can factor the modulus n, they can calculate the private key. This attack is mitigated by using larger prime numbers.

3. **Timing Attacks:** Attackers can measure the time taken to perform certain RSA operations, like modular exponentiation, to deduce information about the private key.

4. **Common Modulus Attack:** If two parties use the same modulus for their RSA keys, and one party's private key is compromised, an attacker can use it to decrypt the other party's messages.

5. **Low Public Exponent Attack:** If a small public exponent (e) is used, and the same plaintext is encrypted multiple times, an attacker can recover the private key.

Outcomes:

CO2: Illustrate different cryptographic algorithms for security.

Conclusion: (Conclusion to be based on the objectives and outcomes achieved)

We can conclude that we have learnt about the RSA cipher for encryption and decryption.

Grade: AA / AB / BB / BC / CC / CD /DD

Signature of faculty in-charge with date

References: Books/ Journals/ Websites:

1. Behrouz A. Forouzan, “Cryptography and Network Security”, Tata McGraw Hill
2. William Stalling, “Cryptography and Network Security”, Prentice Hall