Ishita Hardasmalani

C14-2103058

EXPERIMENT NO.5

Aim: Write a program to implement RSA Algorithm

Theory:

The RSA algorithm, named after its inventors Rivest, Shamir, and Adleman, is a widely used asymmetric cryptographic algorithm. It is commonly used for secure data transmission, digital signatures, and encryption of sensitive information. RSA relies on the mathematical properties of prime numbers and modular arithmetic.

1. Key Generation:

- Choose two distinct prime numbers, p and q.

- Compute their product, n = p * q. This serves as the modulus for both the public and private keys.

- Compute φ(n), where φ is Euler's totient function. For distinct primes, φ(n) = (p - 1)(q - 1).

- Choose an integer e such that 1 < e < φ(n) and e is coprime with φ(n). Typically, e is chosen to be a small prime number, often 65537 (2^16 + 1), because it has a simple binary representation and is relatively prime to most values of φ(n).

- Compute the modular multiplicative inverse of e modulo φ(n), denoted as d. This means finding d such that (d * e) mod φ(n) = 1.

- The public key is (e, n) and the private key is (d, n).

2. Encryption:

- To send a message M, the sender obtains the recipient's public key (e, n).

- The sender converts the message into an integer m such that 0 <= m < n. This can be done using padding schemes if necessary.

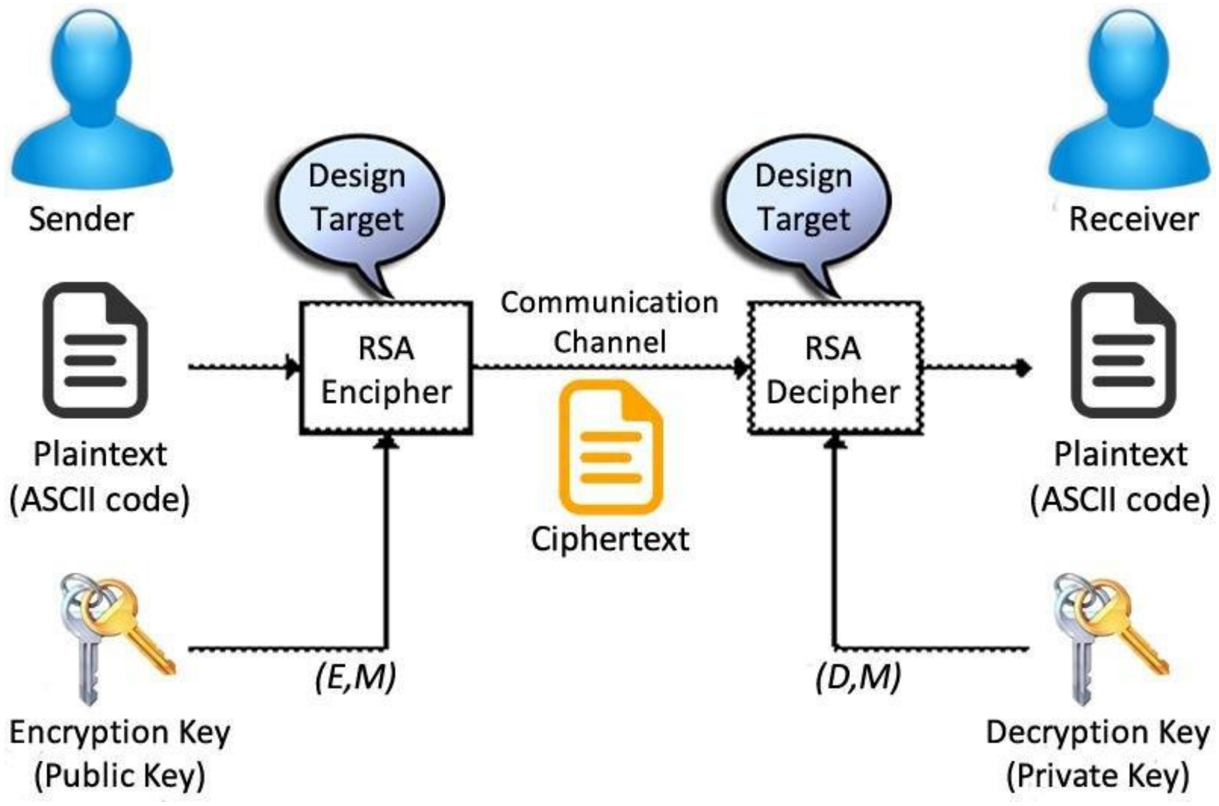
- The sender computes the ciphertext c using the encryption function: c = m^e mod n.

- The sender sends the ciphertext c to the recipient.

3. Decryption:

- The recipient receives the ciphertext c.
- The recipient uses their private key (d, n) to decrypt the ciphertext.
- The recipient computes the original message m using the decryption function: m = c^d mod n.
- The recipient obtains the original message M by reversing any padding schemes applied during encryption.

The security of RSA relies on the practical impossibility of factoring the modulus n into its prime factors given current computing power and algorithms. As such, RSA remains widely used in various cryptographic applications.



The RSA algorithm uses two keys, a public key and a private key, because it's based on asymmetric cryptography. This means that the encryption and decryption keys are different.

1. Encryption with the Public Key:

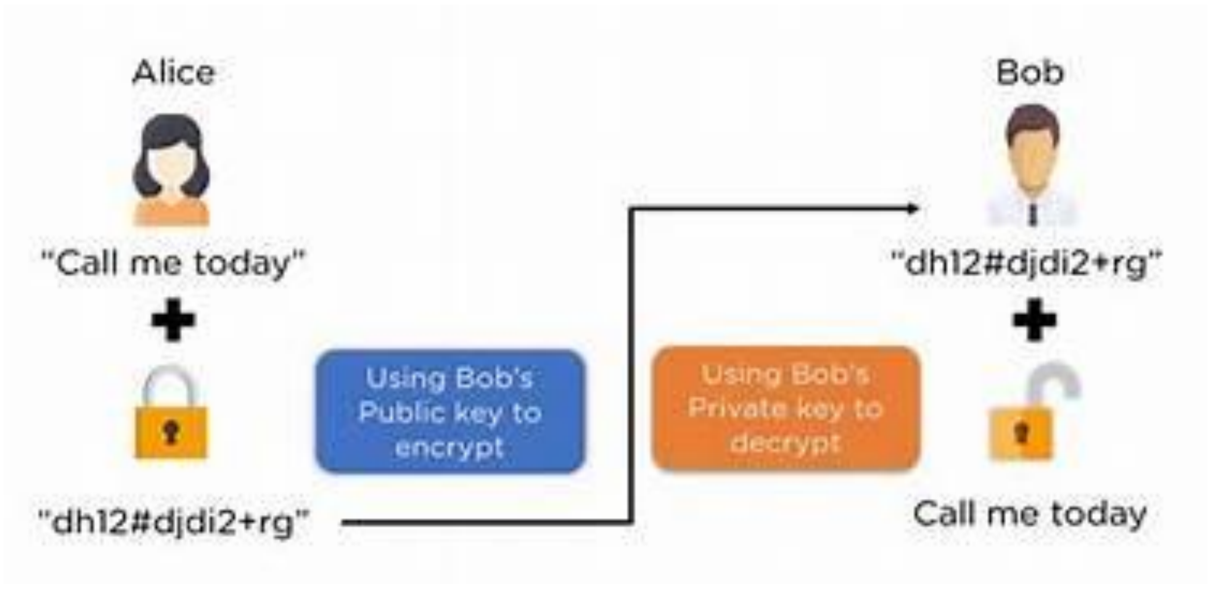
- The public key (e, n) is widely distributed and known to everyone.
- It is used for encryption. Anyone can use the public key to encrypt a message.
- The sender encrypts the message using the recipient's public key and sends the encrypted message.

2. Decryption with the Private Key:

- The private key (d, n) is kept secret and known only to the recipient.
- It is used for decryption. Only the recipient possesses the private key and can decrypt messages encrypted with the corresponding public key.
- The recipient decrypts the received encrypted message using their private key to obtain the original message.

Using two keys enables secure communication because:

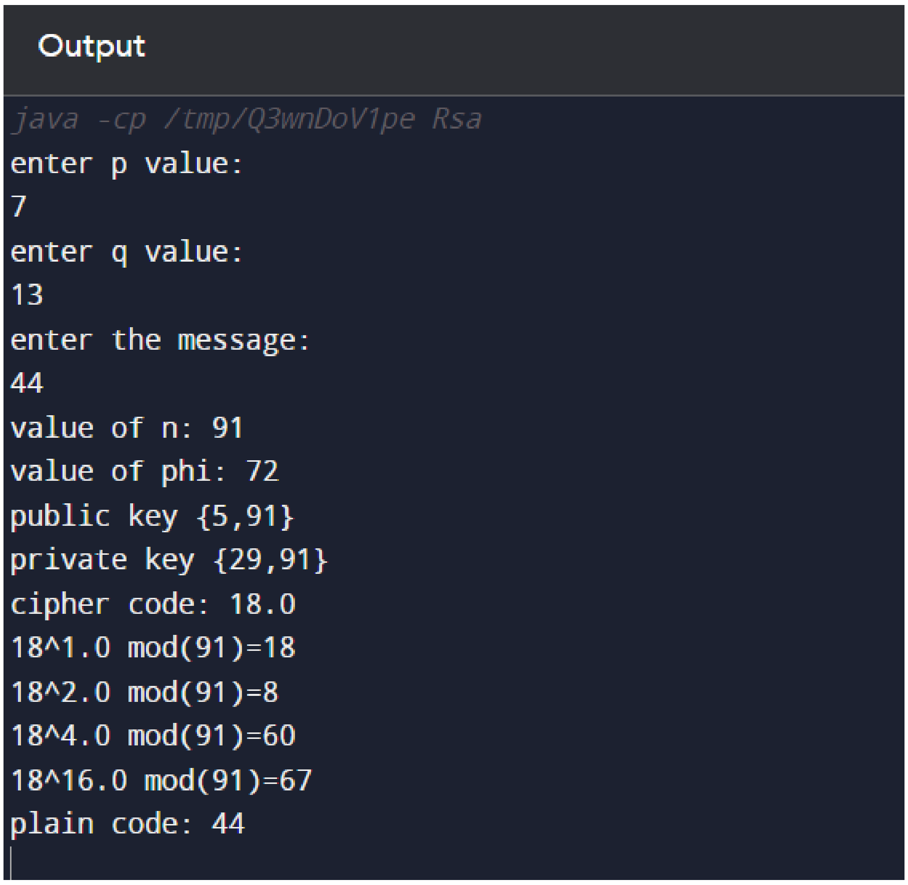
- Encryption with the public key ensures that only the intended recipient, who possesses the corresponding private key, can decrypt the message.
- The private key never needs to be shared or transmitted, reducing the risk of interception or compromise.


Alice
"Call me today"
"dh12#djdi2+rg"
Using Bob's Public key to encrypt
Using Bob's Private key to decrypt
Bob
"dh12#djdi2+rg"
Call me today

Code:

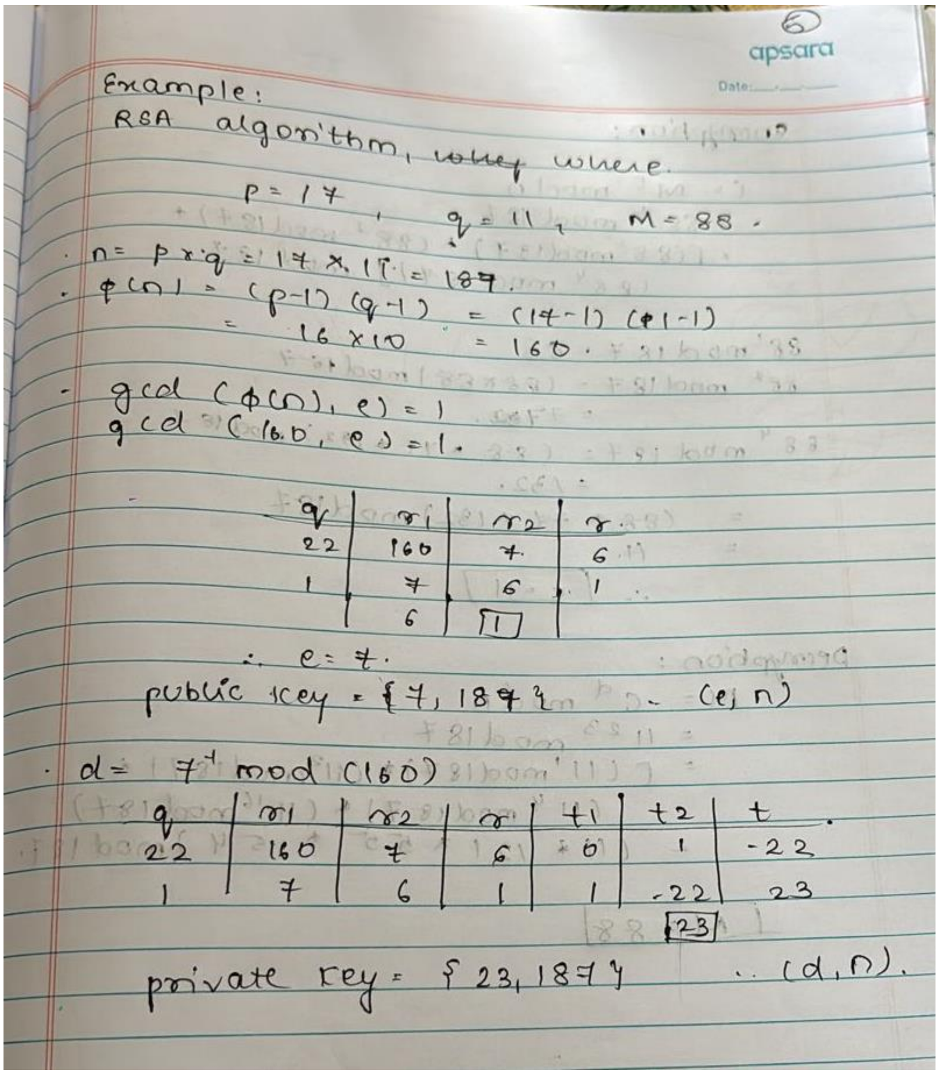
```
import java.io.*;
import java.util.*;
import java.math.*;

public class Rsa {

   static boolean isPrime(int n) {
      if (n <= 1) {
         return false;
      }

      for (int i = 2; i < n; i++) {
         if (n % i == 0) {
            return false;
         }
      }
      return true;
   }

   static int isGcd(int a, int b) {
      if (b == 0) {
         return a;
      }
      return isGcd(b, a % b);
   }

   public static long power(long base, long exponent, long modulus) { // base=11
expo=23 mod=187
      long result = 1;
      long result2 = 1;
      long ogBase = base;
      String binaryExponent = Long.toBinaryString(exponent);
      // base = base % modulus; // 11^1

      while (exponent > 0) {
            if ((exponent & 1) == 1) {

                result = (result * base) % modulus;

            }
            exponent = exponent >> 1;

            base = (base * base) % modulus;

        }

        for (int i = 0; i < binaryExponent.length(); i++) {
            if (binaryExponent.charAt(i) == '1') {
                result2 = (result2 * (int) Math.pow(ogBase, Math.pow(2, i))) % modulus;
                System.out.println(ogBase + "^" + Math.pow(2, i) + " mod(" + (modulus) +
")" + "=" + result2);
            }
            base = (base * base) % modulus;
        }
        return result;
    }

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        int p, q;

        System.out.println("enter p value: ");
        p = sc.nextInt();

        System.out.println("enter q value: ");
        q = sc.nextInt();

        System.out.println("enter the message: ");
        int m = sc.nextInt();

        if ((!isPrime(p) || !isPrime(q)) || (p == q)) {
            System.out.println("enter prime p and q");
        } else {
            long n = p * q;

            System.out.println("value of n: " + n);
            int phi = (p - 1) * (q - 1);
            System.out.println("value of phi: " + phi);
            int e = 2;

            while (e <= phi) {
                if (isGcd(e, phi) == 1) {
                    break;
                } else {
                    e++;
                }
            }

            long d = 2;

            while (((d * e) % phi) != 1) {
                d++;
            }

            System.out.println("public key {" + e + "," + n + "}");
            System.out.println("private key {" + d + "," + n + "}");

            double c = Math.pow(m, e) % n;
            System.out.println("cipher code: " + c);

            long l = (long) c;

            long text = power(l, d, n);
            System.out.println("plain code: " + text);
        }

    }
}
```

Ouput:


Output

```
java -cp /tmp/Q3wnDoV1pe Rsa
enter p value:
7
enter q value:
13
enter the message:
44
value of n: 91
value of phi: 72
public key {5,91}
private key {29,91}
cipher code: 18.0
18^1.0 mod(91)=18
18^2.0 mod(91)=8
18^4.0 mod(91)=60
18^16.0 mod(91)=67
plain code: 44
```

Example:



Example:

RSA algorithm, ~~whey~~ where.

p = 17, q = 11, M = 88.

· $n = p \times q = 17 \times 11 = 187$.

· $\phi(n) = (p-1)(q-1) = (17-1)(11-1)$

$= 16 \times 10 = 160$.

· $\gcd(\phi(n), e) = 1$

$\gcd(160, e) = 1$.

q	r1	r2	r
22	160	7	6
1	7	6	1
	6	$\boxed{1}$	

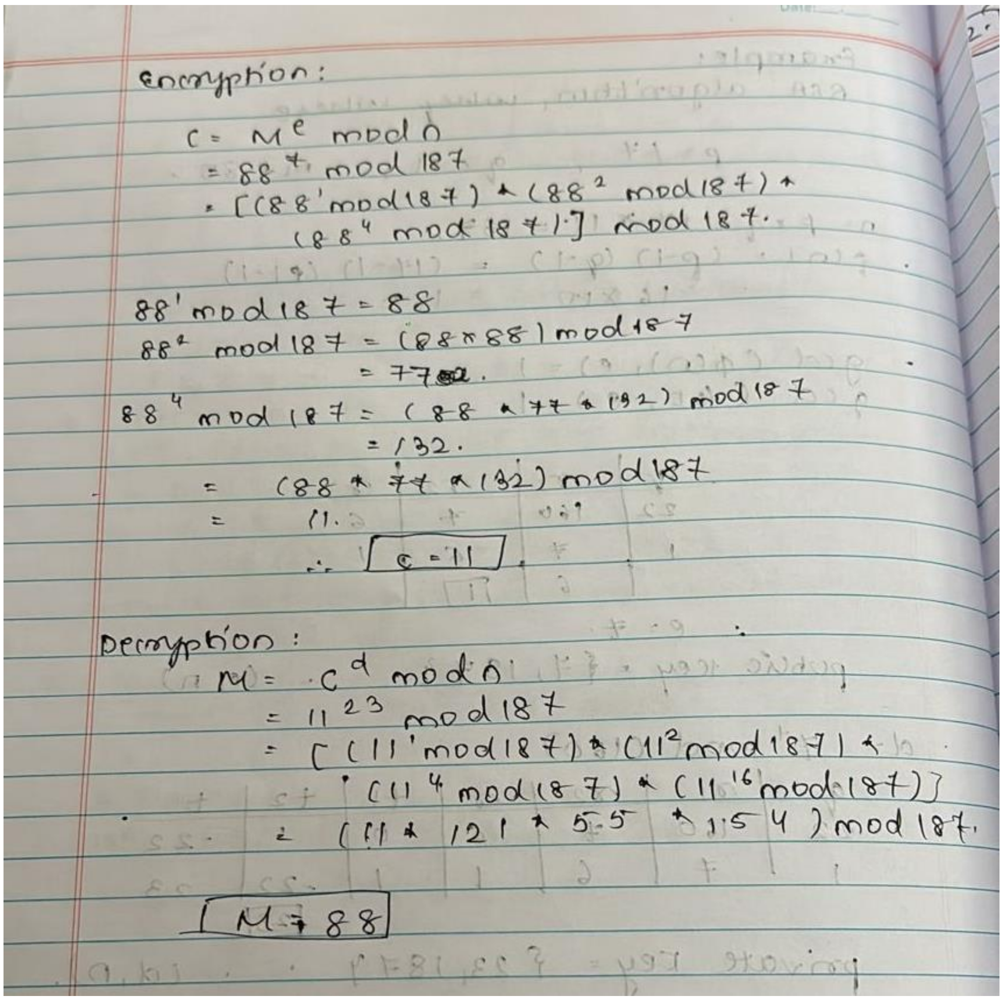
$\therefore e = 7$.

public key = {7, 187} .. (e, n)

· $d = 7^{-1} \bmod (160)$

q	r1	r2	r	t1	t2	t
22	160	7	6	0	1	-22
1	7	6	1	1	-22	23
					$\boxed{23}$	

private key = {23, 187} .. (d, n).



Encryption:

$C = M^e \bmod n$

$= 88^7 \bmod 187$

$= [(88^1 \bmod 187) * (88^2 \bmod 187) * (88^4 \bmod 187)] \bmod 187.$

$88^1 \bmod 187 = 88$

$88^2 \bmod 187 = (88 * 88) \bmod 187$

$= 77.$

$88^4 \bmod 187 = (88 * 77 * 132) \bmod 187$

$= 132.$

$= (88 * 77 * 132) \bmod 187$

$= 11.$

$\therefore \boxed{C = 11}$

Decryption:

$M = C^d \bmod n$

$= 11^{23} \bmod 187$

$= [(11^1 \bmod 187) * (11^2 \bmod 187) * (11^4 \bmod 187) * (11^{16} \bmod 187)]$

$= (11 * 121 * 55 * 154) \bmod 187.$

$\boxed{M = 88}$