**Lab – 4**

**Subject: NIS**

**Aim:** Implement RSA algorithm. Do Key generation 2. Encryption 3. Decryption. Use only Multiply and Square to do Modular Exponentiation. One can use Miller Rabin for Primality Testing (Optional but recommended) . Use large prime numbers due to security concerns

**Program: -**

import java.util.\*;

import java.lang.\*;

class RSA{

    public long gcd(long n1,long n2){

        long gcd=1;

        for(long i = 1; i <= n1 && i <= n2; ++i)

        {

            if(n1 % i==0 && n2 % i==0)

                gcd = i;

        }

        return gcd;

    }

    public static long extendedEuclidian(long a,long n){

        long[] arr = new long[2];

        long r1=n,r2=a,r,t,t1=0,t2=1,gcd,inverse,q;

        while(r2 > 0){

            q=r1/r2;

            r=r1-q\*r2;

            r1=r2;

            r2=r;

            t=t1-q\*t2;

            t1=t2;

            t2=t;

        }

        gcd=r1;

        inverse=t1;

        if(inverse < 0){

            inverse = positiveInvers(inverse,n);

        }

        // arr[0]=inverse;

        // arr[1]=gcd;

        return inverse;

    }

    public static long positiveInvers(long inverse,long n){

        while(inverse < 0){

            inverse = inverse + n;

        }

        return inverse;

    }

    public void RSA\_Encryption(long M,long e,long n){

        System.out.println("Encryption : " + multiplyAndSquare(M,e,n));

    }

    public void RSA\_Decryption(long c,long d,long n){

        System.out.println("Decryption : " + multiplyAndSquare(c,d,n));

    }

    public static long multiplyAndSquare(long a,long X,long n){

        long y=1;

        String x=Long.toBinaryString(X);

        System.out.println("Binary : " + Long.toBinaryString(X));

        for(long i=x.length()-1;i>=0;i--){

            //System.out.println(" B is : " + x.charAt((int)i));

            if(x.charAt((int)i)=='1'){

               y=(y\*a) % n;

            }

            a= (a\*a) % n;

        }

        return y;

    }

    public static void main(String args[]){

        RSA obj1 = new RSA();

        Scanner sc = new Scanner(System.in);

        System.out.println("Enter the plaintext : ");

        long M = sc.nextLong();

        System.out.println("Enetr P and Q :");

        long p = sc.nextLong();

        long q = sc.nextLong();

        long n = p\*q;

        long phi = (p-1) \* (q-1);

        long e=2;

        for(long i=2;i<phi;i++){

            //System.out.println("gcd : " + obj1.gcd(i,phi));

            if(obj1.gcd(i,phi)==1){

                    e=i;

                    break;

            }

        }

        long d = extendedEuclidian(e,phi);

        System.out.println("phi : " + phi + "\ne : " + e + "\nd is : " + d);

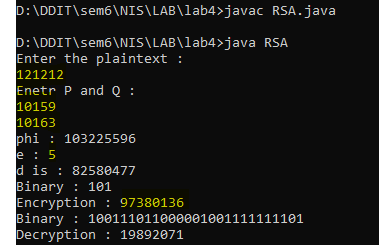
        obj1.RSA\_Encryption(M,e,n);

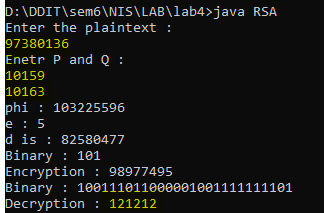
        obj1.RSA\_Decryption(M,d,n);

    }

}

**Output: -**

****

****