Ns 5

Implementation of Symmetric and Asymmetric Algorithm

Message Encryption Using DES Algorithm symmetric

```
import java.security.*;

import java.util.Scanner;

import javax.crypto.*;

import javax.crypto.spec.SecretKeySpec;


public class Main {

    public static void main(String[] args) {

        try {

            Scanner scanner = new Scanner(System.in);

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

            String message = scanner.nextLine();

            KeyGenerator keyGenerator = KeyGenerator.getInstance("DES");

            SecretKey myDesKey = keyGenerator.generateKey();

            Cipher desCipher = Cipher.getInstance("DES/ECB/PKCS5Padding");


            desCipher.init(Cipher.ENCRYPT_MODE, myDesKey);

            byte[] textEncrypted = desCipher.doFinal(message.getBytes());


            System.out.print("Encrypted Message: ");

            for (byte b : textEncrypted) {

                System.out.printf("%02X", b);

            }

            System.out.println();


            desCipher.init(Cipher.DECRYPT_MODE, myDesKey);

            byte[] textDecrypted = desCipher.doFinal(textEncrypted);

            System.out.println("Decrypted Message: " + new String(textDecrypted));

            scanner.close();
        } catch (NoSuchAlgorithmException | NoSuchPaddingException | InvalidKeyException
                | IllegalBlockSizeException | BadPaddingException e) {
            e.printStackTrace();
        }
    }
}
```

Asymmetric

```
import java.math.BigDecimal;
import java.math.BigInteger;
public class ns_5 {
public static void main(String[] args) {
int p, q, n, z, d = 0, e, i;
int msg = 90;
double c;
BigInteger msgback;
p = 43;
q = 59;
n = p * q;
z = (p - 1) * (q - 1);
System.out.println("the value of z = " + z);
for (e = 2; e < z; e++) {
if (gcd(e, z) == 1) {
break;
}
}
System.out.println("the value of e = "+ e);
for (i = 0; i <= 10; i++) {
int x = 1 + (i * z);
if (x % e == 0) {
d = x / e;
break;
}
}
System.out.println("the value of d = "+ d);
c = (Math.pow(msg, e)) % n;
System.out.println("Encrypted message is : "+ c);
BigInteger N = BigInteger.valueOf(n);
BigInteger C = BigDecimal.valueOf(c).toBigInteger();
msgback = (C.pow(d)).mod(N);
System.out.println("Decrypted message is : "+ msgback);
}
static int gcd(int e, int z)
{
if (e == 0)
return z;
else
return gcd(z % e, e);
}
}
```