



CRYPTOGRAPHY & NETWORK SECURITY

Lab Manual



JNTUK – R16

AUTHOR: MADHU BABU JANJANAM

Assoc. Professor, CSE

| S.no | Program | Page no. |
|------|--|----------|
| 1 | To Implement and Break Shift Cipher | 2 |
| 2 | To Implement Mono – Alphabetic Cipher | 4 |
| 3 | To Implement One – Time Pad Cipher | 6 |
| 4 | To Implement Message Authentication Codes (MD5) | 9 |
| 5 | To Implement Cryptographic Hash Function (SHA-256) | 11 |
| 6 | To Implement Symmetric Encryption Cipher DES | 13 |
| 7 | To Implement Symmetric Encryption Cipher AES | 15 |
| 8 | To Implement Diffie – Hellman Key Establishment | 17 |
| 9 | To Implement Public – Key Cryptosystems (RSA) | 20 |
| 10 | To Implement Digital Signatures (DSA) | 22 |

1. Aim: Program to break Shift Cipher

Program:

```
import java.util.*;
class CaesarCipher
{
    public static void main(String args[])
    {
        Scanner sc=new Scanner(System.in);
        int shift,i,n,p,key;
        String str;
        String str1="";
        System.out.println("Enter the Plain Text");
        str=sc.nextLine();
        str=str.toLowerCase();
        n=str.length();
        char ch1[]=str.toCharArray();
        char ch4;
        System.out.println("Enter the value by which each letter of the string is to be shifted");
        shift=sc.nextInt();
        System.out.println();
        System.out.println("Encrypted text is:");
        for(i=0;i<n;i++)
        {
            if(Character.isLetter(ch1[i]))
            {
                ch4=(char)(((int)ch1[i]+shift-97)%26+97);
                str1=str1+ch4;
            }
            else if(ch1[i]==' ')
            {
                str1=str1+ch1[i];
            }
        }
        System.out.println(str1);
        System.out.println("Cipher Text:"+str1);
        n=str1.length();
        char ch2[]=str1.toCharArray();
        char ch3;
        System.out.println();
        System.out.println("Possible Plain text is");
        str1="";
        for(key=26;key>=1;key--)
        {
            for(i=0;i<n;i++)
            {
                if(Character.isLetter(ch2[i]))
                {
```

```

        ch3=(char)((((int)ch2[i]+key-97)%26+97);
        str1=str1+ch3;
    }
    else if(ch2[i]==' ')
    {
        str1=str1+ch2[i];
    }
}
p=26-key;
System.out.println("For Key "+p+"."+str1);
str1="";
}
}
}

```

Output:

```

C:\Users\Madhu\OneDrive\C&NS Lab>java CaesarCipher
Enter the Plain Text
Vasireddy Venkatadri Institute of Technology
Enter the value by which each letter of the string is to be shifted
5

Encrypted text is:
afxnwjiid ajspfyfiwn nsxynyzyj tk yjhmstqtld
Cipher Text:afxnwjiid ajspfyfiwn nsxynyzyj tk yjhmstqtld

Possible Plain text is
For Key 0:afxnwjiid ajspfyfiwn nsxynyzyj tk yjhmstqtld
For Key 1:zewishhc ziroexehvm mrwxmxyxi sj xiglrspskc
For Key 2:ydluhggb yhqndwdgul lqvwlwwh ri whfkqrorjb
For Key 3:xcuktgffa xgpmcvcftk kpukvwvg qh vgejpqnia
For Key 4:wbtjsfeez wfolbubesj jotujuvuf pg ufdiopmphz
For Key 5:vasireddy venkatadri institute of technology
For Key 6:uzrhqdccx udmjzszcqh hmrshstsd ne sdbgmknfx
For Key 7:tyqgpcbbw tcliyrybpg glqrgsrc md rcaflmjmw
For Key 8:sxfobaav sbkhxqaof fkpqfqrqb lc qbzeklildv
For Key 9:rwoenazzu rajgwpwzne ejopepqa kb paydjkhkcu
For Key 10:qvndmzyyt qzifvovymd dinodopoz ja ozxcijgjb
For Key 11:pumclyxxs pyheunxlc chmncnony iz nywbhifias
For Key 12:otlbkxwr oxgdtmtwkb bglmbnmhx hy mxvaghehze
For Key 13:nskajwvvq nwfcslsvja afklalmlw gx lwuzfgdgyq
For Key 14:mrjzivuup mvebrkruiz zejzklkv fw kvtyefcfxp
For Key 15:lqiyhutto ludaqqthy ydijyjkju ev jusxdebewo
For Key 16:kphxgtssn ktczpipsgx xchixijit du itrwcdadvn
For Key 17:jogwfsrrm jsbyohorfw wbgwhihs ct hsqvbczcum
For Key 18:infverqql iraxngnqev vafgvghgr bs grpuabybtl
For Key 19:hmeudqppk hqzwmfmpdu uzefufgfq ar fqotzaxask
For Key 20:gldtcpooj gpyvleloct tydetefep zq epnsyzwzrj
For Key 21:fkcsbonni foxukdnbs sxcdsdedo yp domrxyvyqi
For Key 22:ejbranmmh enwtjcjmar rwbcrdcn xo cnlqwxuxph
For Key 23:diaqzmllg dmvsibilzq qvabqcbm wn bmkpvtwog
For Key 24:chzpylkkf clurhahyp puzapabal vm aljouvsvnf
For Key 25:bgyxkjje bktqgzgjo otyozazk ul zkinturume

```

2. Aim: Program to break Mono-alphabetic cipher

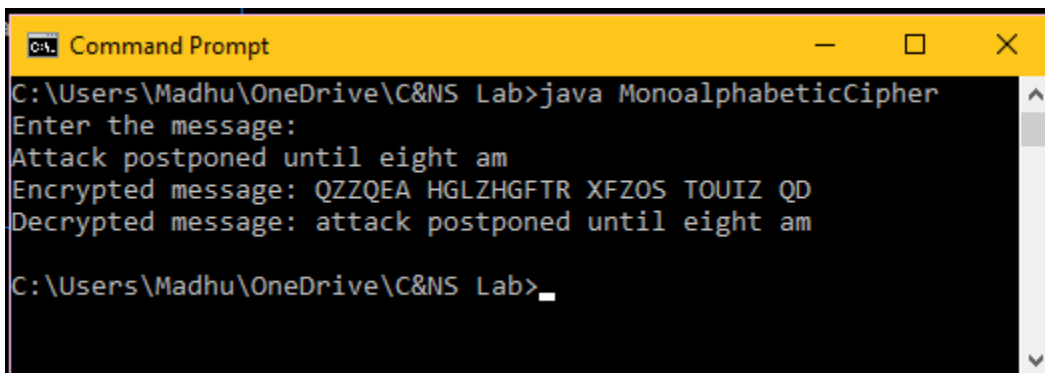
Program:

```
import java.util.Scanner;
public class MonoalphabeticCipher
{
    public static char p[] = { 'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i',
        'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v',
        'w', 'x', 'y', 'z' };
    public static char ch[] = { 'Q', 'W', 'E', 'R', 'T', 'Y', 'U', 'I', 'O',
        'P', 'A', 'S', 'D', 'F', 'G', 'H', 'J', 'K', 'L', 'Z', 'X', 'C',
        'V', 'B', 'N', 'M' };
    static String str;
    public static String doEncryption(String s)
    {
        char c[] = new char[(s.length())];
        for (int i = 0; i < s.length(); i++)
        {
            for (int j = 0; j < 26; j++)
            {
                if (p[j] == s.charAt(i))
                {
                    c[i] = ch[j];
                    break;
                }
            }
        }
        return (new String(c));
    }

    public static String doDecryption(String s)
    {
        char p1[] = new char[(s.length())];
        for (int i = 0; i < s.length(); i++)
        {
            for (int j = 0; j < 26; j++)
            {
                if (ch[j] == s.charAt(i))
                {
                    p1[i] = p[j];
                    break;
                }
            }
        }
        return (new String(p1));
    }

    public static void main(String args[])
    {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the encrypted string:");
        str = sc.nextLine();
        String original = doDecryption(str);
        System.out.println("Original string: " + original);
    }
}
```

```
{
    Scanner sc = new Scanner(System.in);
    System.out.println("Enter the message: ");
    str=sc.next();
    String en = doEncryption(str.toLowerCase());
    System.out.println("Encrypted message: " + en);
    System.out.println("Decrypted message: " + doDecryption(en));
    sc.close();
}
}
```

Output:

```
C:\Users\Madhu\OneDrive\C&NS Lab>java MonoalphabeticCipher
Enter the message:
Attack postponed until eight am
Encrypted message: QZZQEA HGLZHGFR XFZOS TOUIZ QD
Decrypted message: attack postponed until eight am

C:\Users\Madhu\OneDrive\C&NS Lab>
```

3. Aim: Program to implement One-time pad

Program:

```
import java.util.*;
class msg{
    int a=97;
    char all[]=new char[27];
    msg()
    {
        for(int i=0;i<26;i++)
        {
            all[i]=(char)a;
            a++;
        }
    }
    int Ipos(char c)
    {
        int i=0;
        for(;i<26;i++)
        {
            if(all[i]==c)
            {
                break;
            }
        }
        return i;
    }
    char Cpos(int c)
    {
        int i=0;
        for(;i<c;i++)
        {
        }
        return all[i];
    }
}
class OneTimePadCipherImplementation{
String Encryption(String plaintext,String key)
{
    plaintext=plaintext.toLowerCase();
    msg m1=new msg();
    int pt[]=new int[plaintext.length()];
    int k[]=new int[key.length()];
    int ct[]=new int[plaintext.length()];
    for(int i=0;i < plaintext.length();i++)
    {
        pt[i]=m1.Ipos(plaintext.charAt(i));
    }
}
```

```

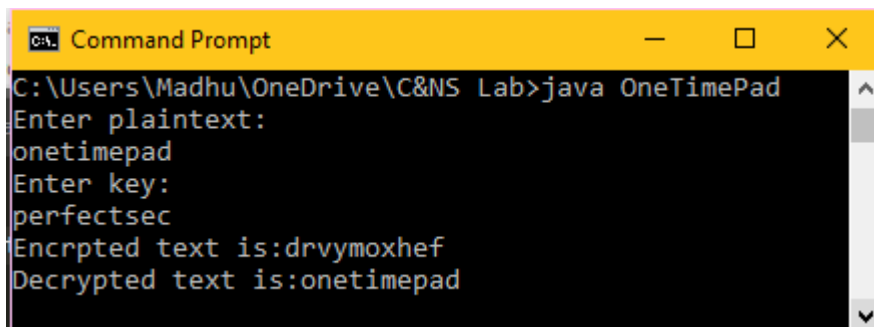
        for(int i=0;i < key.length();i++)
        {
            k[i]=m1.Ipos(key.charAt(i));
        }
        int j=0;
        for(int i=0;i < plaintext.length();i++)
        {
            ct[i]=pt[i]+k[j];
            j++;
            if(j==key.length())
                j=0;
            if(ct[i]>26)
                ct[i]=ct[i]%26;
        }
        String cipher="";
        for(int i=0;i < plaintext.length();i++)
        {
            cipher+=m1.Cpos(ct[i]);
        }
        return cipher;
    }
    String Decryption(String ciphertext,String key)
    {
        String plaintext="";
        msg m1=new msg();
        int pt[]=new int[ciphertext.length()];
        int k[]=new int[key.length()];
        int ct[]=new int[ciphertext.length()];
        for(int i=0;i < ciphertext.length();i++)
        {
            ct[i]=m1.Ipos(ciphertext.charAt(i));
        }
        for(int i=0;i < key.length();i++)
        {
            k[i]=m1.Ipos(key.charAt(i));
        }
        int j=0;
        for(int i=0;i < ciphertext.length();i++)
        {
            pt[i]=ct[i]-k[j];
            j++;
            if(j==key.length())
                j=0;
            if(pt[i] < 0)
                pt[i]+=26;
        }
        String cipher="";
        for(int i=0;i < ciphertext.length();i++)
        {
            plaintext+=m1.Cpos(pt[i]);
        }
    }

```



```
    }  
    return plaintext;  
}  
}  
  
class OneTimePad{  
    public static void main(String args[])throws Exception  
    {  
        String plaintext,key;  
        Scanner scn=new Scanner(System.in);  
        System.out.println("Enter plaintext:");  
        plaintext=scn.nextLine();  
        System.out.println("Enter key:");  
        key=scn.nextLine();  
        OneTimePadCipherImplementation OneTimePad=new  
OneTimePadCipherImplementation();  
        String ciphertext=OneTimePad.Encryption(plaintext,key);  
        System.out.println("Encrypted text is:"+ciphertext);  
        System.out.println("Decrypted text is:"+OneTimePad.Decryption(ciphertext,key));  
    }  
}
```

Output:



```
C:\Users\Madhu\OneDrive\C&NS Lab>java OneTimePad  
Enter plaintext:  
onetimepad  
Enter key:  
perfectsec  
Encrypted text is:drvymoxhef  
Decrypted text is:onetimepad
```

4. Aim: Program to implement Message authentication codes (MD5)

Program:

```
import java.math.BigInteger;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;

// Java program to calculate MD5 hash value
public class MD5 {
    public static String getMd5(String input)
    {
        try {

            // Static getInstance method is called with hashing MD5
            MessageDigest md = MessageDigest.getInstance("MD5");

            // digest() method is called to calculate message digest
            // of an input digest() return array of byte
            byte[] messageDigest = md.digest(input.getBytes());

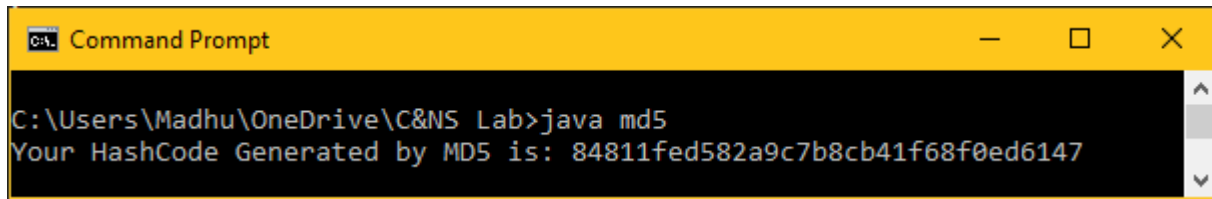
            // Convert byte array into signum representation
            BigInteger no = new BigInteger(1, messageDigest);

            // Convert message digest into hex value
            String hashtext = no.toString(16);
            while (hashtext.length() < 32) {
                hashtext = "0" + hashtext;
            }
            return hashtext;

        }

        // For specifying wrong message digest algorithms
        catch (NoSuchAlgorithmException e) {
            throw new RuntimeException(e);
        }
    }

    // Driver code
    public static void main(String args[]) throws NoSuchAlgorithmException
    {
        String s = "VVIT";
        System.out.println("Your HashCode Generated by MD5 is: " + getMd5(s));
    }
}
```

Output:

```
Command Prompt
C:\Users\Madhu\OneDrive\C&NS Lab>java md5
Your HashCode Generated by MD5 is: 84811fed582a9c7b8cb41f68f0ed6147
```

5. Aim: Program to implement Cryptographic Hash Function (SHA-256)

Program:

```
import java.math.BigInteger;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;

// Java program to calculate MD5 hash value
public class SHA {
    public static String getSHA(String input)
    {
        try {

            // Static getInstance method is called with hashing MD5
            MessageDigest hash = MessageDigest.getInstance("SHA-256");

            // digest() method is called to calculate message digest
            // of an input digest() return array of byte
            byte[] messageDigest = hash.digest(input.getBytes());

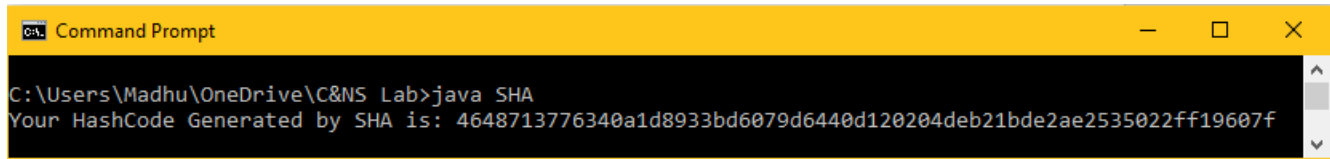
            // Convert byte array into signum representation
            BigInteger no = new BigInteger(1, messageDigest);

            // Convert message digest into hex value
            String hashtext = no.toString(16);
            while (hashtext.length() < 32) {
                hashtext = "0" + hashtext;
            }
            return hashtext;

        }

        // For specifying wrong message digest algorithms
        catch (NoSuchAlgorithmException e) {
            throw new RuntimeException(e);
        }
    }

    // Driver code
    public static void main(String args[]) throws NoSuchAlgorithmException
    {
        String s = "VVIT";
        System.out.println("Your HashCode Generated by SHA is: " + getSHA(s));
    }
}
```

Output:A screenshot of a Windows Command Prompt window. The title bar is yellow and says "Command Prompt". The window has standard Windows window controls (minimize, maximize, close) on the right. The command prompt shows the following text:
C:\Users\Madhu\OneDrive\C&NS Lab>java SHA
Your HashCode Generated by SHA is: 4648713776340a1d8933bd6079d6440d120204deb21bde2ae2535022ff19607f
The text is white on a black background. There is a small icon in the top left corner of the command prompt window.

6. Aim: Program to implement DES Symmetric Encryption

Program:

```
import javax.crypto.Cipher;
import javax.crypto.KeyGenerator;
import javax.crypto.SecretKey;
import java.util.Base64;

class DesEncrypter {
    Cipher ecipher;
    Cipher dcipher;

    DesEncrypter(SecretKey key) throws Exception {
        ecipher = Cipher.getInstance("DES");
        dcipher = Cipher.getInstance("DES");
        ecipher.init(Cipher.ENCRYPT_MODE, key);
        dcipher.init(Cipher.DECRYPT_MODE, key);
    }

    public String encrypt(String str) throws Exception {
        // Encode the string into bytes using utf-8
        byte[] utf8 = str.getBytes("UTF8");

        // Encrypt
        byte[] enc = ecipher.doFinal(utf8);

        // Encode bytes to base64 to get a string
        //return new sun.misc.BASE64Encoder().encode(enc);
        return Base64.getEncoder().encodeToString(enc);
    }

    public String decrypt(String str) throws Exception {
        // Decode base64 to get bytes
        //byte[] dec = new sun.misc.BASE64Decoder().decodeBuffer(str);
        byte[] dec = Base64.getDecoder().decode(str);
        byte[] utf8 = dcipher.doFinal(dec);

        // Decode using utf-8
        return new String(utf8, "UTF8");
    }
}

public class DES {
    public static void main(String[] argv) throws Exception {
        SecretKey key = KeyGenerator.getInstance("DES").generateKey();
        DesEncrypter encrypter = new DesEncrypter(key);
        String encrypted = encrypter.encrypt("Don't tell anybody!");
        System.out.println(encrypted);
    }
}
```

```
String decrypted = encrypter.decrypt(encrypted);  
System.out.println(decrypted);  
}  
}
```

Output:



A screenshot of a Windows Command Prompt window with a yellow title bar labeled "Command Prompt". The window contains the following text:

```
C:\Users\Madhu\OneDrive\C&NS Lab>java DES  
6jcTBwTRHrIotkZQVpc8uxDen6Y9vkcq  
Don't tell anybody!
```

7. Aim: Program to implement AES Symmetric Encryption

Program:

```
import javax.crypto.Cipher;
import javax.crypto.KeyGenerator;
import javax.crypto.SecretKey;
import java.util.Base64;

class AesEncrypter {
    Cipher ecipher;

    Cipher dcipher;

    AesEncrypter(SecretKey key) throws Exception {
        ecipher = Cipher.getInstance("AES");
        dcipher = Cipher.getInstance("AES");
        ecipher.init(Cipher.ENCRYPT_MODE, key);
        dcipher.init(Cipher.DECRYPT_MODE, key);
    }

    public String encrypt(String str) throws Exception {
        // Encode the string into bytes using utf-8
        byte[] utf8 = str.getBytes("UTF8");

        // Encrypt
        byte[] enc = ecipher.doFinal(utf8);

        // Encode bytes to base64 to get a string
        //return new sun.misc.BASE64Encoder().encode(enc);
        return Base64.getEncoder().encodeToString(enc);
    }

    public String decrypt(String str) throws Exception {
        // Decode base64 to get bytes
        //byte[] dec = new sun.misc.BASE64Decoder().decodeBuffer(str);
        byte[] dec = Base64.getDecoder().decode(str);

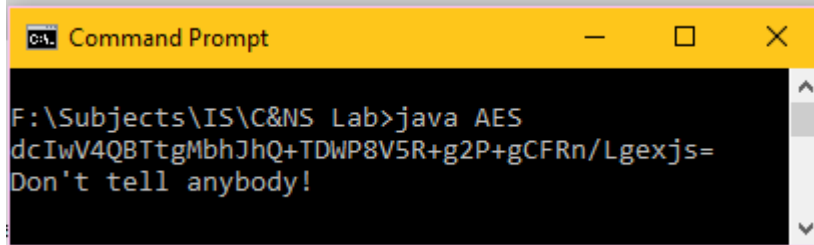
        byte[] utf8 = dcipher.doFinal(dec);

        // Decode using utf-8
        return new String(utf8, "UTF8");
    }
}

public class AES {
    public static void main(String[] argv) throws Exception {
        SecretKey key = KeyGenerator.getInstance("AES").generateKey();
        AesEncrypter encrypter = new AesEncrypter(key);
    }
}
```



```
String encrypted = encrypter.encrypt("Don't tell anybody!");  
System.out.println(encrypted);  
String decrypted = encrypter.decrypt(encrypted);  
System.out.println(decrypted);  
}  
}
```

Output:A screenshot of a Windows Command Prompt window with a yellow title bar labeled "Command Prompt". The window shows the execution of a Java program. The prompt is "F:\Subjects\IS\C&NS Lab>". The command entered is "java AES". The output displayed is "dcIwV4QBTtgMbhJhQ+TDWP8V5R+g2P+gCFRn/Lgexjs=" followed by a new line and "Don't tell anybody!".

```
F:\Subjects\IS\C&NS Lab>java AES  
dcIwV4QBTtgMbhJhQ+TDWP8V5R+g2P+gCFRn/Lgexjs=  
Don't tell anybody!
```

8. Aim: Program to implement Diffie – Hellman Key Establishment

Program:

DHServer.java

```
import java.net.*;
import java.io.*;

public class DHServer {
    public static void main(String[] args) throws IOException
    {
        try {
            int port = 8088;
            // Server Key
            int b = 3;
            // Client p, g, and key
            double clientP, clientG, clientA, B, Bdash;
            String Bstr;

            // Established the Connection
            ServerSocket serverSocket = new ServerSocket(port);
            System.out.println("Waiting for client on port " + serverSocket.getLocalPort() +
" ...");

            Socket server = serverSocket.accept();
            System.out.println("Just connected to " + server.getRemoteSocketAddress());

            // Server's Private Key
            System.out.println("From Server : Private Key = " + b);

            // Accepts the data from client
            DataInputStream in = new DataInputStream(server.getInputStream());

            clientP = Integer.parseInt(in.readUTF()); // to accept p
            System.out.println("From Client : P = " + clientP);

            clientG = Integer.parseInt(in.readUTF()); // to accept g
            System.out.println("From Client : G = " + clientG);

            clientA = Double.parseDouble(in.readUTF()); // to accept A
            System.out.println("From Client : Public Key = " + clientA);

            B = ((Math.pow(clientG, b)) % clientP); // calculation of B
            Bstr = Double.toString(B);

            // Sends data to client
            // Value of B
            OutputStream outToClient = server.getOutputStream();
            DataOutputStream out = new DataOutputStream(outToClient);
```

```

        out.writeUTF(Bstr); // Sending B
        Bdash = ((Math.pow(clientA, b)) % clientP); // calculation of Bdash
        System.out.println("Secret Key to perform Symmetric Encryption = "
                           + Bdash);

        server.close();
    }
    catch (SocketTimeoutException s) {
        System.out.println("Socket timed out!");
    }
    catch (IOException e) {
    }
}
}

```

DHClient.java

```

import java.net.*;
import java.io.*;

public class DHClient {
    public static void main(String[] args)
    {
        try {
            String pstr, gstr, Astr;
            String serverName = "localhost";
            int port = 8088;

            // Declare p, g, and Key of client
            int p = 23;
            int g = 9;
            int a = 4;
            double Adash, serverB;

            // Established the connection
            System.out.println("Connecting to " + serverName
                              + " on port " + port);
            Socket client = new Socket(serverName, port);
            System.out.println("Just connected to "
                              + client.getRemoteSocketAddress());

            // Sends the data to client
            OutputStream outToServer = client.getOutputStream();
            DataOutputStream out = new DataOutputStream(outToServer);

            pstr = Integer.toString(p);
            out.writeUTF(pstr); // Sending p

            gstr = Integer.toString(g);
            out.writeUTF(gstr); // Sending g

```

```
double A = ((Math.pow(g, a)) % p); // calculation of A
Astr = Double.toString(A);
out.writeUTF(Astr); // Sending A

// Client's Private Key
System.out.println("From Client : Private Key = " + a);

// Accepts the data
DataInputStream in = new DataInputStream(client.getInputStream());

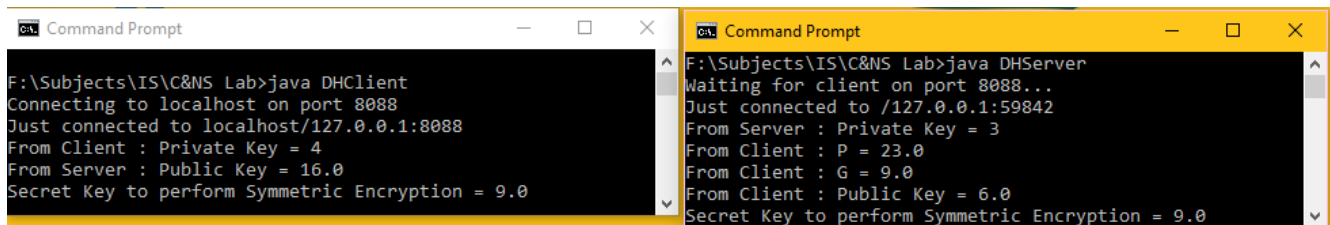
serverB = Double.parseDouble(in.readUTF());
System.out.println("From Server : Public Key = " + serverB);

Adash = ((Math.pow(serverB, a)) % p); // calculation of Adash

System.out.println("Secret Key to perform Symmetric Encryption = "
                    + Adash);

client.close();
}
catch (Exception e) {
    e.printStackTrace();
}
}
```

Output:



```
Command Prompt
F:\Subjects\IS\C&NS Lab>java DHClient
Connecting to localhost on port 8088
Just connected to localhost/127.0.0.1:8088
From Client : Private Key = 4
From Server : Public Key = 16.0
Secret Key to perform Symmetric Encryption = 9.0

Command Prompt
F:\Subjects\IS\C&NS Lab>java DHServer
Waiting for client on port 8088...
Just connected to /127.0.0.1:59842
From Server : Private Key = 3
From Client : P = 23.0
From Client : G = 9.0
From Client : Public Key = 6.0
Secret Key to perform Symmetric Encryption = 9.0
```

9. Aim: Program to implement Public Key Cryptosystems (RSA)

Program:

```
import java.math.BigInteger;
import java.security.SecureRandom;

public class RSADemo {
    private final static BigInteger one = new BigInteger("1");
    private final static SecureRandom random = new SecureRandom();

    private BigInteger privateKey;
    private BigInteger publicKey;
    private BigInteger modulus;

    // generate an N-bit (roughly) public and private key
    RSADemo(int N) {
        BigInteger p = BigInteger.probablePrime(N/2, random);
        BigInteger q = BigInteger.probablePrime(N/2, random);
        BigInteger phi = (p.subtract(one)).multiply(q.subtract(one));
        System.out.println("prime p = " + p);
        System.out.println("prime q = " + q);

        modulus = p.multiply(q);
        System.out.println("phi = " + phi);
        publicKey = new BigInteger("65537"); // common value in practice = 2^16 + 1
        privateKey = publicKey.modInverse(phi);
    }

    BigInteger encrypt(BigInteger message) {
        return message.modPow(publicKey, modulus);
    }

    BigInteger decrypt(BigInteger encrypted) {
        return encrypted.modPow(privateKey, modulus);
    }

    public String toString() {
        String s = "";
        s += "public = " + publicKey + "\n";
        s += "private = " + privateKey + "\n";
        s += "modulus = " + modulus;
        return s;
    }

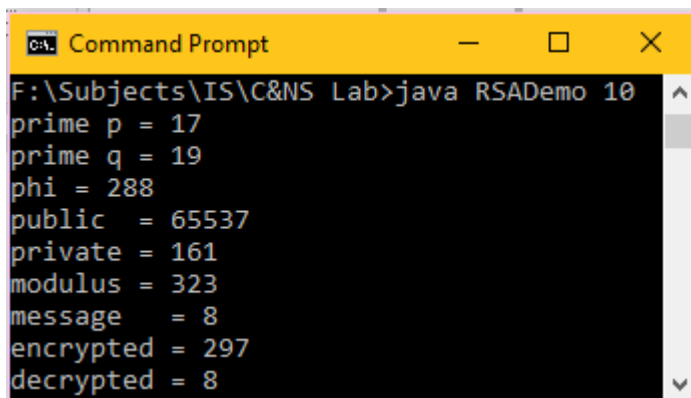
    public static void main(String[] args) {
        int N = Integer.parseInt(args[0]);
        RSADemo key = new RSADemo(N);
        System.out.println(key);
    }
}
```

```
// create random message, encrypt and decrypt
BigInteger message = new BigInteger("8");

//// create message by converting string to integer
// String s = "test";
// byte[] bytes = s.getBytes();
// BigInteger message = new BigInteger(bytes);

BigInteger encrypt = key.encrypt(message);
BigInteger decrypt = key.decrypt(encrypt);
System.out.println("message = " + message);
System.out.println("encrypted = " + encrypt);
System.out.println("decrypted = " + decrypt);
}
}
```

Output:



```
C:\> Command Prompt
F:\Subjects\IS\C&NS Lab>java RSADemo 10
prime p = 17
prime q = 19
phi = 288
public = 65537
private = 161
modulus = 323
message = 8
encrypted = 297
decrypted = 8
```

10. Aim: Program to Implement Digital Signatures (DSS)

Program:

```
import java.security.KeyPair;
import java.security.KeyPairGenerator;
import java.security.PrivateKey;
import java.security.PublicKey;
import java.security.Signature;
import java.util.Scanner;
import java.io.*;

public class CreatingDigitalSignature {
    public static void main(String args[]) throws Exception {
        //Accepting text from user
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter some text");
        String msg = sc.nextLine();

        //Creating KeyPair generator object
        KeyPairGenerator keyPairGen = KeyPairGenerator.getInstance("DSA");

        //Initializing the key pair generator
        keyPairGen.initialize(2048);

        //Generate the pair of keys
        KeyPair pair = keyPairGen.generateKeyPair();

        //Getting the private key from the key pair
        PrivateKey privKey = pair.getPrivate();
        PublicKey pubKey = pair.getPublic();
        System.out.println(privKey);
        System.out.println(pubKey);

        //Creating a Signature object
        Signature sign = Signature.getInstance("SHA256withDSA");

        //Initialize the signature
        sign.initSign(privKey);
        byte[] bytes = "msg".getBytes();

        //Adding data to the signature
        sign.update(bytes);

        //Calculating the signature
        byte[] signature = sign.sign();

        //Printing Signature to console
        for(int i=0;i<signature.length;i++)
```

```

        System.out.print(" "+signature[i]);
    System.out.println();

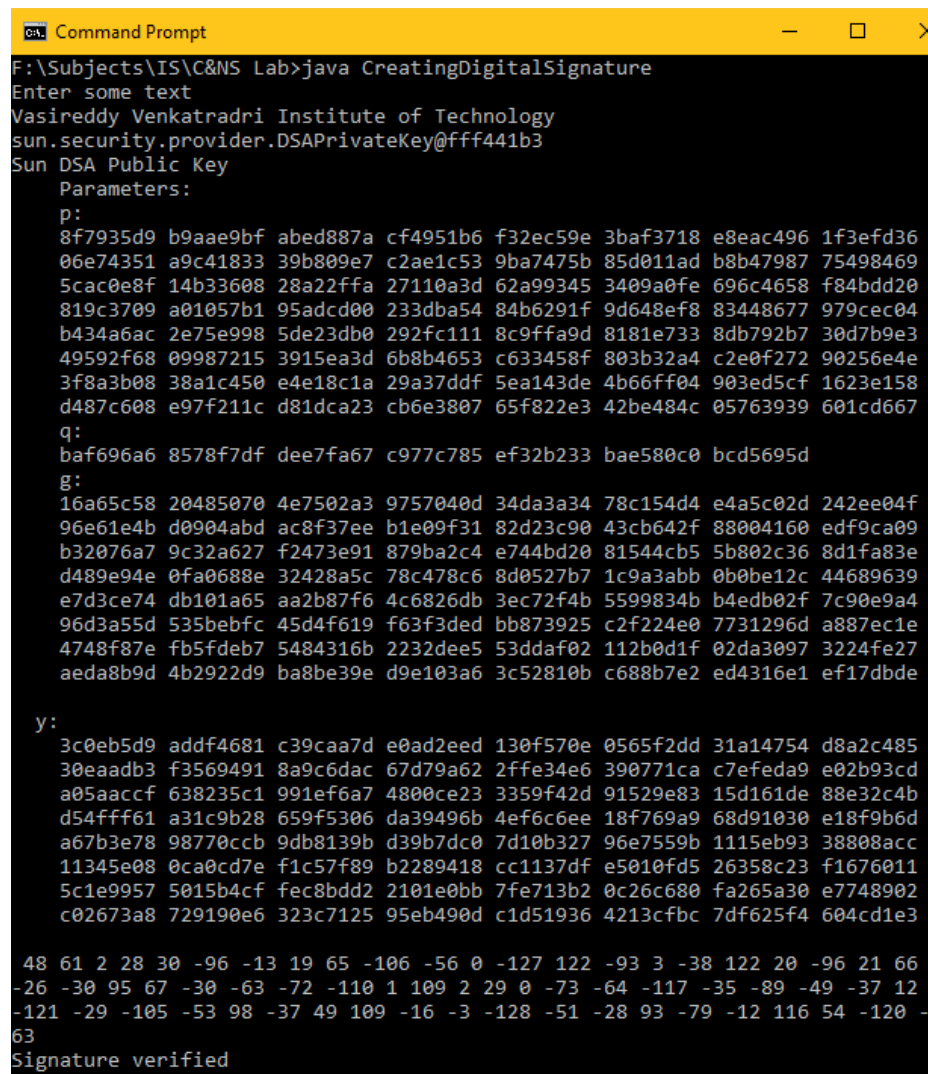
    //Initializing the signature verification
    sign.initVerify(pair.getPublic());
    sign.update(bytes);

    //Verifying the signature
    boolean bool = sign.verify(signature);

    if(bool) {
        System.out.println("Signature verified");
    } else {
        System.out.println("Signature failed");
    }
}
}
}

```

Output:



```

C:\> java CreatingDigitalSignature
Enter some text
Vasireddy Venkatradri Institute of Technology
sun.security.provider.DSAPrivateKey@fff441b3
Sun DSA Public Key
Parameters:
p:
8f7935d9 b9aae9bf abed887a cf4951b6 f32ec59e 3baf3718 e8eac496 1f3efd36
06e74351 a9c41833 39b809e7 c2ae1c53 9ba7475b 85d011ad b8b47987 75498469
5cac0e8f 14b33608 28a22ffa 27110a3d 62a99345 3409a0fe 696c4658 f84bdd20
819c3709 a01057b1 95adcd00 233dba54 84b6291f 9d648ef8 83448677 979cec04
b434a6ac 2e75e998 5de23db0 292fc111 8c9ffa9d 8181e733 8db792b7 30d7b9e3
49592f68 09987215 3915ea3d 6b8b4653 c633458f 803b32a4 c2e0f272 90256e4e
3f8a3b08 38a1c450 e4e18c1a 29a37ddf 5ea143de 4b66ff04 903ed5cf 1623e158
d487c608 e97f211c d81dca23 cb6e3807 65f822e3 42be484c 05763939 601cd667
q:
baf696a6 8578f7df dee7fa67 c977c785 ef32b233 bae580c0 bcd5695d
g:
16a65c58 20485070 4e7502a3 9757040d 34da3a34 78c154d4 e4a5c02d 242ee04f
96e61e4b d0904abd ac8f37ee b1e09f31 82d23c90 43cb642f 88004160 edf9ca09
b32076a7 9c32a627 f2473e91 879ba2c4 e744bd20 81544cb5 5b802c36 8d1fa83e
d489e94e 0fa0688e 32428a5c 78c478c6 8d0527b7 1c9a3abb 0b0be12c 44689639
e7d3ce74 db101a65 aa2b87f6 4c6826db 3ec72f4b 5599834b b4edb02f 7c90e9a4
96d3a55d 535bebfc 45d4f619 f63f3ded bb873925 c2f224e0 7731296d a887ec1e
4748f87e fb5fdeb7 5484316b 2232dee5 53ddaf02 112b0d1f 02da3097 3224fe27
aeda8b9d 4b2922d9 ba8be39e d9e103a6 3c52810b c688b7e2 ed4316e1 ef17dbde

y:
3c0eb5d9 addf4681 c39caa7d e0ad2eed 130f570e 0565f2dd 31a14754 d8a2c485
30eaadb3 f3569491 8a9c6dac 67d79a62 2ffe34e6 390771ca c7efeda9 e02b93cd
a05aaccf 638235c1 991ef6a7 4800ce23 3359f42d 91529e83 15d161de 88e32c4b
d54fff61 a31c9b28 659f5306 da39496b 4ef6c6ee 18f769a9 68d91030 e18f9b6d
a67b3e78 98770ccb 9db8139b d39b7dc0 7d10b327 96e7559b 1115eb93 38808acc
11345e08 0ca0cd7e f1c57f89 b2289418 cc1137df e5010fd5 26358c23 f1676011
5c1e9957 5015b4cf fec8bdd2 2101e0bb 7fe713b2 0c26c680 fa265a30 e7748902
c02673a8 729190e6 323c7125 95eb490d c1d51936 4213cfbc 7df625f4 604cd1e3

48 61 2 28 30 -96 -13 19 65 -106 -56 0 -127 122 -93 3 -38 122 20 -96 21 66
-26 -30 95 67 -30 -63 -72 -110 1 109 2 29 0 -73 -64 -117 -35 -89 -49 -37 12
-121 -29 -105 -53 98 -37 49 109 -16 -3 -128 -51 -28 93 -79 -12 116 54 -120 -
63
Signature verified

```