# Department of Computer Science

Gujarat University

## Certificate

Roll No: 37 Seat No: ____

This is to certify that Mr./Ms. Jeel Digeshkumar Mavani student of MCA Semester – V has duly completed his/her term work for the semester ending in December 2020, in the subject of Network Security towards partial fulfillment of his/her Degree of Masters in Computer Applications.

**11th, December 2020**
Date of Submission

Internal Faculty

Head of Department

D E P A R T M E N T  O F  C O M P U T E R  S C I E N C E
R O L L W A L A  C O M P U T E R  C E N T R E
G U J A R A T  U N I V E R S I T Y
**M.C.A. –5**

R O L L NO : 37
N A M E : Jeel Mavani
S U B J E C T : Network Security

| NO. | TITLE | PAGE NO. | DATE | SIGN |
|---|---|---|---|---|
| **1.** | Caesar Cipher | **4** | **13-Aug-2020** | |
| **2** | Caesar Cipher Client Server | **5** | **15-Aug-2020** | |
| **3** | Substitution Cipher | **7** | **23-Aug-2020** | |
| **4** | Transposition Cipher | **8** | **30-Aug-2020** | |
| **5** | One Time Pad | **10** | **2-Sept-2020** | |
| **6** | PBox | **11** | **10-Sept-2020** | |
| **7** | SBox | **12** | **14-Sept-2020** | |
| **8** | MD5 | **13** | **17-Sept-2020** | |
| **9** | MD5Utils | **14** | **20-Sept-2020** | |
| **10** | RSA | **15** | **25-Sept-2020** | |
| **11.** | RSA Encryption | **17** | **15-Oct-2020** | |
| **12** | SHA | **18** | **20-Oct-2020** | |
| **13** | AES | **19** | **12-Nov-2020** | |
| **14** | DES | **20** | **v-2020** | |
| **15** | Implement authentication Service. The sender sends password in encrypted format to the receiver,<br><br>The receiver checks the password (after decrypting and applying hash) in its database/array and replies.<br><br>Back as success or failure. (Note: Here the password is stored as hash in database). | **24** | **27-Nov-2020** | |

D E P A R T M E N T  O F  C O M P U T E R  S C I E N C E
R O L L W A L A  C O M P U T E R  C E N T R E
G U J A R A T  U N I V E R S I T Y
**M.C.A. –5**

R O L L NO : 37
N A M E : Jeel Mavani
S U B J E C T : Network Security

| | | | | |
|---|---|---|---|---|
| 16 | Implement a firewall that behaves like forwarder. It does not forward the packet that contains the word “terrorist”. | **28** | **27-Nov-2020** | |
| 17 | Implement NAT functionality. The NAT works like forwarder, that will forward to appropriate receiver. | **30** | **29th-Nov-2020** | |
| **18** | Key Distribution<br><br>Implement a program to demonstrate the functioning of a KDC. There are three entities: sender, receiver and KDC. Assume that Sender and Receiver have already established their own individual permanent secret keys with KDC. The sender requests the KDC to issue a session key to communicate with receiver. The KDC is supposed to give session key information to sender in a secure way. The same session key is also to be communicated to the receiver securely. Use a suitable protocol to achieve the above functionality. | **33** | **29th-Nov-2020** | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

**Name: Jeel Mavani**
**Roll No: 37**
**Subject: Network Security**
**Sem: MCA-5**

## Caeser_Cipher.java

```
import java.util.Scanner;
class Caesar_Cipher{
	Scanner sc;
	String plain_text,cipher_text,decrypted_text;
	Caesar_Cipher(){
		sc=new Scanner(System.in);
		cipher_text="";
		decrypted_text="";
	}
	void getData(){
		System.out.println("Enter Data : ");
		plain_text=sc.nextLine();
		plain_text=plain_text.trim();
	}
	void encryptData(){
		char ch;
		for(int i=0;i<plain_text.length();i++){
			ch=plain_text.charAt(i);
			if(ch>='A' && ch<='Z'){
				ch=(char)(ch+3);
				if(ch>'Z'){
					ch=(char)(ch-'Z'+'A'-1);
				}
				cipher_text=cipher_text+ch;
			}
			else if(ch>='a' && ch<='z'){
				ch=(char)(ch+3);
				if(ch>'z'){
					ch=(char)(ch-'z'+'a'-1);
				}
				cipher_text=cipher_text+ch;
			}
		}
		System.out.println("Cipher Text  :  "+cipher_text);
	}
	void decryptData(){
		char ch;
		for(int i=0;i<cipher_text.length();i++){
```

```
                        ch=cipher_text.charAt(i);
                        if(ch>='A' && ch<='Z'){
                                ch=(char)(ch-3);
                                if(ch<'A'){
                                        ch=(char)(ch-'A'+'Z'+1);
                                }
                                decrypted_text=decrypted_text+ch;
                        }
                        else if(ch>='a' && ch<='z'){
                                ch=(char)(ch+3);
                                if(ch<'a'){
                                        ch=(char)(ch-'a'+'z'+1);
                                }
                                decrypted_text=decrypted_text+ch;
                        }
                }
                System.out.println("Decrypted Text  :  "+decrypted_text);
        }
        public static void main(String args[]){
                Caesar_Cipher cc=new Caesar_Cipher();
                cc.getData();
                cc.encryptData();
                cc.decryptData();
        }
}
```

---

## Caeser Cipher Through Client-Server

---

Caeser_Cipher_Client.java

```
import java.util.Scanner;
import java.net.*;
import java.io.*;
class Caeser_Cipher_Client{
        static void encrypt_send(String msg) throws Exception{
                Socket s=new Socket(InetAddress.getLocalHost(),1234);
                DataOutputStream out=new DataOutputStream(s.getOutputStream());
                String cipher_text="";
                char ch;
                for(int i=0;i<msg.length();i++){
                        ch=msg.charAt(i);
                        if(ch>='A' && ch<='Z'){
                                ch=(char)(ch+3);
                                if(ch>'Z'){
                                        ch=(char)(ch-'Z'+'A'-1);
                                }
                                cipher_text=cipher_text+ch;
                        }
                        else if(ch>='a' && ch<='z'){
```

```
                    ch=(char)(ch+3);
                    if(ch>'z'){
                        ch=(char)(ch-'z'+'a'-1);
                    }
                    cipher_text=cipher_text+ch;
                }
            }
            out.writeUTF(cipher_text);
        }
        public static void main(String args[]) throws Exception{
            Scanner sc=new Scanner(System.in);
            System.out.println("Enter Data");
            String msg=sc.nextLine();
            encrypt_send(msg);
        }
}
```

Caeser_Cipher_Server.java

```
import java.util.Scanner;
import java.net.*;
import java.io.*;
class Caeser_Cipher_Server{
        ServerSocket ss;
        DataInputStream in;
        Caeser_Cipher_Server() throws Exception{
            ss=new ServerSocket(1234);
        }
        void decrypt_print() throws Exception{
            System.out.println("Server");
            Socket s=ss.accept();
            in=new DataInputStream(s.getInputStream());
            String cipher_text=in.readUTF();
            System.out.println("Recieved Data : " + cipher_text);
            String decrypted_text="";
            char ch;
            for(int i=0;i<cipher_text.length();i++){
                ch=cipher_text.charAt(i);
                if(ch>='A' && ch<='Z'){
                    ch=(char)(ch-3);
                    if(ch<'A'){
                        ch=(char)(ch-'A'+'Z'+1);
                    }
                    decrypted_text=decrypted_text+ch;
                }
                else if(ch>='a' && ch<='z'){
                    ch=(char)(ch+3);
                    if(ch<'a'){
                        ch=(char)(ch-'a'+'z'+1);
                    }
                    decrypted_text=decrypted_text+ch;
                }
```

```
		}
		System.out.println("Decrypted Text  :  "+decrypted_text);
	}
	public static void main(String args[]) throws Exception{
		Caeser_Cipher_Server c=new Caeser_Cipher_Server();
		c.decrypt_print();
	}
}
```

## Substitution Cipher

Substitution_Cipher.java

```
import java.util.Scanner;
class Substitution_Cipher{
	Scanner sc;
	String plain_text,cipher_text,decrypted_text;
	int shift;
	Substitution_Cipher(){
		sc=new Scanner(System.in);
		cipher_text="";
		decrypted_text="";
		shift=0;
	}
	void getData(){
		System.out.println("Enter Data : ");
		plain_text=sc.nextLine();
		plain_text=plain_text.trim();
		System.out.println("Enter Shift : ");
		shift=sc.nextInt();
	}
	void encryptData(){
		char ch;
		for(int i=0;i<plain_text.length();i++){
			ch=plain_text.charAt(i);
			if(ch>='A' && ch<='Z'){
				ch=(char)(ch+shift);
				if(ch>'Z'){
					ch=(char)(ch-'Z'+'A'-1);
				}
				cipher_text=cipher_text+ch;
			}
			else if(ch>='a' && ch<='z'){
				ch=(char)(ch+shift);
				if(ch>'z'){
					ch=(char)(ch-'z'+'a'-1);
				}
				cipher_text=cipher_text+ch;
				}
			}
			System.out.println("Cipher Text  :  "+cipher_text);
		}
		void decryptData(){
			char ch;
			for(int i=0;i<cipher_text.length();i++){
				ch=cipher_text.charAt(i);
				if(ch>='A' && ch<='Z'){
					ch=(char)(ch-shift);
					if(ch<'A'){
						ch=(char)(ch-'A'+'Z'+1);
					}
					decrypted_text=decrypted_text+ch;
				}
				else if(ch>='a' && ch<='z'){
					ch=(char)(ch+shift);
					if(ch<'a'){
						ch=(char)(ch-'a'+'z'+1);
					}
					decrypted_text=decrypted_text+ch;
				}
			}
			System.out.println("Decrypted Text  :  "+decrypted_text);
		}
		public static void main(String args[]){
			Substitution_Cipher cc=new Substitution_Cipher();
			cc.getData();
			cc.encryptData();
			cc.decryptData();
		}
}
```

---

## Transposition Cipher

---

Transposition_Cipher.java

```
import java.util.*;
class transposition
{
		public static void main(String args[])throws Exception
		{
			Scanner sc=new Scanner(System.in);
			System.out.println("enter key(of unique alphabets):");
			String k=sc.nextLine();
			char[] key=k.toCharArray();
			char[] temp_key=new char[key.length];
			System.arraycopy(key,0,temp_key,0,key.length);
			Arrays.sort(temp_key);
```

```
System.out.print("\nenter string :");
String t=sc.nextLine();
char[] str=t.toCharArray();
for(int i=0;i<str.length;i++)
{
        if(str[i]==' ')
                str[i]='$';
}
int index=0,row;
if(((str.length)%(key.length))==0)
        row=((str.length)/(key.length));
else
        row=((str.length)/(key.length))+1;
char[] cipher=new char[(row*(key.length))];
int ci=0;
while(ci<(row*(key.length)))
{
        for(int i=0;i<key.length;i++)
        {
        index=0;
        for(int j=0;j<key.length;j++)
        {
                if(temp_key[i]==key[j])
                {
                        index=j;
                        int l=0;
                        while(l<row)
                        {
                                if(index<str.length)
                                {
                                cipher[ci]=str[index];
                                ci++;
                                l++;
                                index=index+(key.length);
                                }
                                else
                                {
                                        cipher[ci]='!';
                                        ci++;
                                        l++;
                                }
                        }
                        break;
                }
        }
        }
}
System.out.println("Cipher text:");
for(int i=0;i<cipher.length;i++)
{
        System.out.print(cipher[i]);
}
```

```
                char[] decipher=new char[cipher.length];
                int di=0;
                int l=0;
                while(di<cipher.length)
                {
                        for(int i=0;i<key.length;i++)
                        {
                        index=0;
                                for(int j=0;j<key.length;j++)
                                {
                                        if(key[i]==temp_key[j])
                                        {
                                                index=((j)*row)+l;
                                                decipher[di]=cipher[index];
                                                if(decipher[di]=='$')
                                                        decipher[di]=' ';
                                                if(decipher[di]=='!')
                                                        decipher[di]='\0';
                                                di++;
                                                break;
                                        }
                                }
                        }
                        l++;
                }
                System.out.println("\ndecipher text:");
                for(int i=0;i<cipher.length;i++)
                {
                        System.out.print(decipher[i]);
                }
        }
}
```

---

## OneTimePad

---

### OneTimePad.java

```
import java.util.*;
class OneTimePad{
        static String generate_key(String msg){
                String key="";
                Random str_ascii=new Random();
                for(int i=0;i<msg.length();i++){
                        int n=str_ascii.nextInt(26);
                        key=key+(char)(n+97);
                }
                return key;
        }
        static String encryption(String msg,String k){
```

```
		String enc_text="";
		for(int i=0;i<msg.length();i++){
			char enc_val=(char)(msg.charAt(i)^k.charAt(i));
			enc_text=enc_text+enc_val;
		}
		return enc_text;
	}
	static String decryption(String msg,String k){
		String dec_text="";
		for(int i=0;i<msg.length();i++){
			char dec_val=(char)(msg.charAt(i)^k.charAt(i));
			dec_text=dec_text+dec_val;
		}
		return dec_text;
	}
	public static void main(String[]args){
		Scanner s=new Scanner(System.in);
		System.out.print("Enter your string::-");
		String message=s.nextLine();
		String key=generate_key(message);
		System.out.println("Key Used For Process : " + key);
		String encrypted_msg=encryption(message,key);
		System.out.println("Your encrypted message is ::"+encrypted_msg);
		String decrypted_msg=decryption(encrypted_msg,key);
		System.out.println("Your decrypted message is ::"+decrypted_msg);

	}
}
```

## PBox

P_Box.java

```
import java.util.Scanner;
class P_Box{
	public String doEncryption(String s){
		byte p[]=new byte[8];
		byte pTemp[]=new byte[8];
		pTemp=s.getBytes();
		p[0]=pTemp[4];
		p[1]=pTemp[0];
		p[2]=pTemp[5];
		p[3]=pTemp[7];
		p[4]=pTemp[1];
		p[5]=pTemp[3];
		p[6]=pTemp[2];
		p[7]=pTemp[6];
		return(new String(p));
	}
	public String doDecryption(String s){
		byte p[]=new byte[8];
		byte pTemp[]=new byte[8];
		pTemp=s.getBytes();
		p[0]=pTemp[1];
		p[1]=pTemp[4];
		p[2]=pTemp[6];
		p[3]=pTemp[5];
		p[4]=pTemp[0];
		p[5]=pTemp[2];
		p[6]=pTemp[7];
		p[7]=pTemp[3];
		return(new String(p));
	}
	public static void main(String args[]){
		Scanner sc=new Scanner(System.in);
		System.out.println("Enter String(Only 8 Characters) : ");
		String plaintext=sc.nextLine();
		P_Box p_box=new P_Box();
		System.out.println("Encrypted Text : " + p_box.doEncryption(plaintext));
		System.out.println("Decrypted Text : " +
p_box.doDecryption(p_box.doEncryption(plaintext)));
	}
}
```

---

## SBox

---

S_Box.java

```
import java.util.*;
class S_Box{
	char key[][];
	Random r;
	S_Box(){
		r=new Random();
		int add=r.nextInt(5);
		key=new char[52][2];
		char temp='A',ch;
		for(int i=0;i<key.length;i++,temp++){
			if(temp<='Z' && temp>='A'){
				ch=(char)(temp+add);
				if(ch>'Z'){
					ch=(char)(ch-'Z'+'A'-1);
				}
				key[i][0]=(char)temp;
				key[i][1]=(char)(ch);
				if(temp=='Z'){
					temp=(char)('a'-1);
				}
```

```
                    }
                    else if(temp<='z' && temp>='a'){
                              ch=(char)(temp+add);
                              if(ch>'z'){
                                        ch=(char)(ch-'z'+'a'-1);
                              }
                              key[i][0]=(char)temp;
                              key[i][1]=(char)(ch);
                    }
          }
     }
     public String doEncryption(String s){
          String cipherText="";
          for(int i=0;i<s.length();i++){
                    for(int j=0;j<key.length;j++){
                              if(s.charAt(i)==key[j][0]){
                                        cipherText+=key[j][1];
                              }
                    }
          }
          return cipherText;
     }
     public void doDecryption(String s){
          String plainText="";
          for(int i=0;i<s.length();i++){
                    for(int j=0;j<key.length;j++){
                              if(s.charAt(i)==key[j][1]){
                                        plainText+=key[j][0];
                              }
                    }
          }
          System.out.println("Decrypted Text : " + plainText);
     }
     public static void main(String args[]){
          S_Box s=new S_Box();
          Scanner sc=new Scanner(System.in);
          System.out.println("Enter Message : ");
          String plaintext=sc.nextLine();
          String encrypted = s.doEncryption(plaintext);
          System.out.println("Encrypted Text : " + encrypted);
          s.doDecryption(encrypted);
     }
}
```

---

## MD5

---

MD5.java

```
import java.util.Scanner;
import java.math.*;
import java.security.*;
class MD5{
        private String doEncryption(String text) throws Exception{
                MessageDigest md=MessageDigest.getInstance("MD5");
                byte[] msg=md.digest(text.getBytes());
                BigInteger bigInt=new BigInteger(1,msg);
                String hashValue=bigInt.toString(16);
                while(hashValue.length()<32)
                        hashValue+=0+hashValue;
                return hashValue;
        }
        public static void main(String args[]) throws Exception{
                MD5 MD5=new MD5();
                Scanner sc=new Scanner(System.in);
                System.out.println("Enter Message : ");
                String text=sc.nextLine();
                System.out.println("Hash Text : " + MD5.doEncryption(text));
        }
}
```

---

## MD5Utils

---

MD5Utils.java

```
import java.nio.charset.Charset;
import java.nio.charset.StandardCharsets;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;

public class MD5Utils {

   private static final Charset UTF_8 = StandardCharsets.UTF_8;
   private static final String OUTPUT_FORMAT = "%-20s:%s";

   private static byte[] digest(byte[] input) {
      MessageDigest md;
      try {
         md = MessageDigest.getInstance("MD5");
      } catch (NoSuchAlgorithmException e) {
         throw new IllegalArgumentException(e);
      }
      byte[] result = md.digest(input);
      return result;
   }

   private static String bytesToHex(byte[] bytes) {
      StringBuilder sb = new StringBuilder();
      for (byte b : bytes) {
         sb.append(String.format("%02x", b));
```

```
        }
        return sb.toString();
    }

    public static void main(String[] args) {

        String pText = "A quick brown fox jumps over  the lazy dog who was a Java Programmer";
        System.out.println(String.format(OUTPUT_FORMAT, "Input (string)", pText));
        System.out.println(String.format(OUTPUT_FORMAT, "Input (length)", pText.length()));

        byte[] md5InBytes = MD5Utils.digest(pText.getBytes(UTF_8));
        System.out.println(String.format(OUTPUT_FORMAT, "MD5 (hex) ",
bytesToHex(md5InBytes)));
        // fixed length, 16 bytes, 128 bits.
        System.out.println(String.format(OUTPUT_FORMAT, "MD5 (length)", md5InBytes.length));

    }

}
```

---

## RSA

---

RSADemo.java

```
import java.io.DataInputStream;
import java.io.IOException;
import java.math.BigInteger;
import java.util.Random;
public class RSADemo{
    private BigInteger P;
    private BigInteger Q;
    private BigInteger N;
    private BigInteger PHI;
    private BigInteger e;
    private BigInteger d;
    private int maxLength = 1024;
    private Random R;
    public RSADemo(){
        R = new Random();
        P = BigInteger.probablePrime(maxLength, R);
         Q = BigInteger.probablePrime(maxLength, R);
        N = P.multiply(Q);
       PHI = P.subtract(BigInteger.ONE).multiply(  Q.subtract(BigInteger.ONE));
        e = BigInteger.probablePrime(maxLength / 2, R);
        while (PHI.gcd(e).compareTo(BigInteger.ONE) > 0 && e.compareTo(PHI) < 0)
        {
            e.add(BigInteger.ONE);
        }
        d = e.modInverse(PHI);
    }

    public RSADemo(BigInteger e, BigInteger d, BigInteger N)
    {
        this.e = e;
        this.d = d;
        this.N = N;
    }

    public static void main (String [] arguments) throws IOException
    {
        RSADemo rsa = new RSADemo();
        DataInputStream input = new DataInputStream(System.in);
        String inputString;
        System.out.println("Enter message you wish to send.");
        inputString = input.readLine();
        System.out.println("Encrypting the message: " + inputString);
        System.out.println("The message in bytes is:: "
                + bToS(inputString.getBytes()));
        // encryption
        byte[] cipher = rsa.encryptMessage(inputString.getBytes());
        // decryption
        byte[] plain = rsa.decryptMessage(cipher);
        System.out.println("Decrypting Bytes: " + bToS(plain));
        System.out.println("Plain message is: " + new String(plain));
    }

    private static String bToS(byte[] cipher)
    {
        String temp = "";
        for (byte b : cipher)
        {
            temp += Byte.toString(b);
        }
        return temp;
    }

    // Encrypting the message
    public byte[] encryptMessage(byte[] message)
    {
        return (new BigInteger(message)).modPow(e, N).toByteArray();
    }

    // Decrypting the message
    public byte[] decryptMessage(byte[] message)
    {
        return (new BigInteger(message)).modPow(d, N).toByteArray();
    }
}
```

# RSAEncryption

RSAEncryption.java

```
import java.security.KeyPair;
import java.security.KeyPairGenerator;
import java.security.PrivateKey;
import java.security.PublicKey;
import java.util.Base64;

import javax.crypto.Cipher;

public class RSAEncryption
{
   static String plainText = "Plain text which need to be encrypted by Java RSA Encryption in ECB
Mode";

   public static void main(String[] args) throws Exception
   {
      // Get an instance of the RSA key generator
      KeyPairGenerator keyPairGenerator = KeyPairGenerator.getInstance("RSA");
      keyPairGenerator.initialize(4096);

      // Generate the KeyPair
      KeyPair keyPair = keyPairGenerator.generateKeyPair();

      // Get the public and private key
      PublicKey publicKey = keyPair.getPublic();
      PrivateKey privateKey = keyPair.getPrivate();

      System.out.println("Original Text  : "+plainText);

      // Encryption
      byte[] cipherTextArray = encrypt(plainText, publicKey);
      String encryptedText = Base64.getEncoder().encodeToString(cipherTextArray);
      System.out.println("Encrypted Text : "+encryptedText);

      // Decryption
      String decryptedText = decrypt(cipherTextArray, privateKey);
      System.out.println("DeCrypted Text : "+decryptedText);
   }

   public static byte[] encrypt (String plainText,PublicKey publicKey ) throws Exception
   {
      //Get Cipher Instance RSA With ECB Mode and OAEPWITHSHA-512ANDMGF1PADDING
Padding
      Cipher cipher = Cipher.getInstance("RSA/ECB/OAEPWITHSHA-
512ANDMGF1PADDING");
```

```
        //Initialize Cipher for ENCRYPT_MODE
        cipher.init(Cipher.ENCRYPT_MODE, publicKey);

        //Perform Encryption
        byte[] cipherText = cipher.doFinal(plainText.getBytes()) ;

        return cipherText;
    }

    public static String decrypt (byte[] cipherTextArray, PrivateKey privateKey) throws Exception
    {
        //Cipher Instance RSA With ECB Mode and OAEPWITHSHA-512ANDMGF1PADDING
        Cipher cipher = Cipher.getInstance("RSA/ECB/RSA/ECB/OAEPWITHSHA-
512ANDMGF1PADDING");

        //Initialize Cipher for DECRYPT_MODE
        cipher.init(Cipher.DECRYPT_MODE, privateKey);

        //Perform Decryption
        byte[] decryptedTextArray = cipher.doFinal(cipherTextArray);

        return new String(decryptedTextArray);
    }
}
```

---

## SHA

---

SHA.java

```
import java.util.Scanner;
import java.math.*;
import java.security.*;
class SHA{
        private String doEncryption(String text) throws Exception{
                MessageDigest md=MessageDigest.getInstance("SHA-1");
                byte[] msg=md.digest(text.getBytes());
                BigInteger bigInt=new BigInteger(1,msg);
                String hashValue=bigInt.toString(16);
                while(hashValue.length()<32)
                        hashValue+=0+hashValue;
                return hashValue;
        }
        public static void main(String args[]) throws Exception{
                SHA sha=new SHA();
                Scanner sc=new Scanner(System.in);
                System.out.println("Enter Message : ");
                String text=sc.nextLine();
                System.out.println("Hash Text : " + sha.doEncryption(text));
```

```
	}
}
```

## AES

AES.java

```
import javax.crypto.*;
import javax.crypto.spec.*;
import java.util.Scanner;
class AES{
	private byte[] key;
	AES(){
		key="kHFlksfddsaKHBDS".getBytes();
	}
	private byte[] doEncryption(String plainText) throws Exception{
		SecretKeySpec secretKey=new SecretKeySpec(key,"AES");
		Cipher cipher=Cipher.getInstance("AES");
		cipher.init(Cipher.ENCRYPT_MODE,secretKey);
		return cipher.doFinal(plainText.getBytes());
	}
	private byte[] doDecryption(String cipherText) throws Exception{
		SecretKeySpec secretKey=new SecretKeySpec(key,"AES");
		Cipher cipher=Cipher.getInstance("AES");
		cipher.init(Cipher.DECRYPT_MODE,secretKey);
		return cipher.doFinal(cipherText.getBytes());
	}
	public static void main(String args[]) throws Exception{
		Scanner sc=new Scanner(System.in);
		System.out.println("Enter Message : ");
		String plainText=sc.nextLine();
		AES aes=new AES();
		String cipherText=new String(aes.doEncryption(plainText));
		System.out.println("Encrypted Text : " + cipherText);
		System.out.println("Encrypted Text : " + new String(aes.doDecryption(cipherText)));
	}
}
```

## DES

DES.java

```
import javax.crypto.*;
import javax.crypto.spec.*;
import java.util.Scanner;
```

```
class DES{
	private SecretKey secretKey;
	DES() throws Exception{
		secretKey=KeyGenerator.getInstance("DES").generateKey();
	}
	private byte[] doEncryption(String plainText) throws Exception{

		Cipher cipher=Cipher.getInstance("DES");
		cipher.init(Cipher.ENCRYPT_MODE,secretKey);
		return cipher.doFinal(plainText.getBytes());
	}
	private byte[] doDecryption(String cipherText) throws Exception{
		Cipher cipher=Cipher.getInstance("DES");
		cipher.init(Cipher.DECRYPT_MODE,secretKey);
		return cipher.doFinal(cipherText.getBytes());
	}
	public static void main(String args[]) throws Exception{
		Scanner sc=new Scanner(System.in);
		System.out.println("Enter Message : ");
		String plainText=sc.nextLine();
		DES DES=new DES();
		String cipherText=new String(DES.doEncryption(plainText));
		System.out.println("Encrypted Text : " + cipherText);
		System.out.println("Encrypted Text : " + new
String(DES.doDecryption(cipherText)));
	}
}
```

---

## Encrytion

---

prg1_client.java

```
import java.io.*;
import java.util.*;
import java.net.*;
class prg1_client
{
	public static void main(String args[])throws Exception
	{
		Scanner sc=new Scanner(System.in);
		System.out.print("Enter the username");
		String uname=sc.nextLine();
		System.out.print("Enter the password");
		String pwd=sc.nextLine();
		Socket s=new Socket("127.0.0.1",5678);
		DataOutputStream dos=new DataOutputStream(s.getOutputStream());
		dos.writeUTF(uname);
		conversion c=new conversion();
		String encrypt_pwd=c.convert((3*-1),pwd);
		dos.writeUTF(encrypt_pwd);
		DataInputStream dis=new DataInputStream(s.getInputStream());
		String msg=dis.readUTF();
		System.out.println(msg);
		s.close();
	}
}
class conversion
{
	char temp,t;
	int asci=0;
	public String convert(int key,String inputmsg)
	{
		String op="";
		int min=0,max=0,flag=0;
		for(int i=0;i<inputmsg.length();i++)
		{
			temp=inputmsg.charAt(i);
			asci=temp+key;
			if (temp>=97&&temp<=122)
			{

				min=97;
				max=122;
				flag=1;
			}
			else if(temp>=48&&temp<=57)
			{
				min=48;
				max=57;
				flag=1;
			}
			else if(temp>=65&&temp<=90)
			{
				min=65;
				max=90;
				flag=1;
			}
			else
				flag=0;

			if(flag==1)
			{
				if(asci>max)
				{
					int rem=asci-max;
					t=(char)((min-1)+rem);
				}
				else if(asci<min)
				{
					int rem=min-asci;
					t=(char)((max+1)-rem);
```

```
                        }
                        else
                        {
                                t=(char)(asci);
                        }
                        op=op+t;
                }
                else
                {
                        op=op+temp;
                }
        }
        return op;
    }
}
```

prg1_server.java

```
import java.util.*;
import java.io.*;
import java.net.*;
import java.sql.*;
class prg1_server
{

    public static void main(String args[])throws Exception
    {
        ServerSocket ss=new ServerSocket(5678);
        Socket s=ss.accept();
        DataInputStream dis=new DataInputStream(s.getInputStream());
        String uname=dis.readUTF();
        String e_password=dis.readUTF();
        System.out.println(uname);
        System.out.println(e_password);
        conversion c=new conversion();
        String d_password=c.convert(3,e_password);

        Connection con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/user","root","");

        Statement st= con.createStatement();

        String sql="select password from user_details where user_id='"+uname+"';";

        ResultSet rs=st.executeQuery(sql);
        rs.next();
        String password=rs.getString(1);
        con.close();
        DataOutputStream dos=new DataOutputStream(s.getOutputStream());
        if(password.equals(d_password))
        {
            dos.writeUTF("logged in successfully!");
				}
				else
				{
					dos.writeUTF("wrong password!!");
				}
		}
}

class conversion
{
		char temp,t;
		int asci=0;
		public String convert(int key,String inputmsg)
		{
				String op="";
				int min=0,max=0,flag=0;
				for(int i=0;i<inputmsg.length();i++)
				{
					temp=inputmsg.charAt(i);
					asci=temp+key;
					if (temp>=97&&temp<=122)
					{

						min=97;
						max=122;
						flag=1;
					}
					else if(temp>=48&&temp<=57)
					{
						min=48;
						max=57;
						flag=1;
					}
					else if(temp>=65&&temp<=90)
					{
						min=65;
						max=90;
						flag=1;
					}
					else
						flag=0;

					if(flag==1)
					{
						if(asci>max)
						{
							int rem=asci-max;
							t=(char)((min-1)+rem);
						}
						else if(asci<min)
						{
							int rem=min-asci;
					t=(char)((max+1)-rem);
				}
				else
				{
					t=(char)(asci);
				}
				op=op+t;
			}
			else
			{
				op=op+temp;
			}
		}
		return op;
	}
}
```

## Firewall-1

prg2_client.java

```
import java.io.*;
import java.util.*;
import java.net.*;
class prg2_client
{
	public static void main(String args[])throws Exception
	{
		Scanner sc=new Scanner(System.in);
		System.out.print("Enter the username");
		String uname=sc.nextLine();
		System.out.print("Enter the password");
		String pwd=sc.nextLine();
		Socket s=new Socket("127.0.0.1",5678);
		DataOutputStream dos=new DataOutputStream(s.getOutputStream());
		dos.writeUTF(uname);
		conversion c=new conversion();
		String encrypt_pwd=c.convert((3*-1),pwd);
		dos.writeUTF(encrypt_pwd);
		DataInputStream dis=new DataInputStream(s.getInputStream());
		String msg=dis.readUTF();
		System.out.println(msg);
		s.close();
	}
}
class conversion
{
	char temp,t;
	int asci=0;
```

```
public String convert(int key,String inputmsg)
{
        String op="";
        int min=0,max=0,flag=0;
        for(int i=0;i<inputmsg.length();i++)
        {
                temp=inputmsg.charAt(i);
                asci=temp+key;
                if (temp>=97&&temp<=122)
                {

                        min=97;
                        max=122;
                        flag=1;
                }
                else if(temp>=48&&temp<=57)
                {
                        min=48;
                        max=57;
                        flag=1;
                }
                else if(temp>=65&&temp<=90)
                {
                        min=65;
                        max=90;
                        flag=1;
                }
                else
                        flag=0;

                if(flag==1)
                {
                        if(asci>max)
                        {
                                int rem=asci-max;
                                t=(char)((min-1)+rem);
                        }
                        else if(asci<min)
                        {
                                int rem=min-asci;
                                t=(char)((max+1)-rem);
                        }
                        else
                        {
                                t=(char)(asci);
                        }
                        op=op+t;
                }
                else
                {
                        op=op+temp;
                }
```

```
			}
			return op;
		}
}
```

prg2_server.java

```
import java.util.*;
import java.io.*;
import java.net.*;
import java.math.*;
import java.security.*;
import java.sql.*;
class prg2_server
{

		public static void main(String args[])throws Exception
		{
			ServerSocket ss=new ServerSocket(5678);
			Socket s=ss.accept();
			DataInputStream dis=new DataInputStream(s.getInputStream());
			String uname=dis.readUTF();
			String e_password=dis.readUTF();
			conversion c=new conversion();
			String d_password=c.convert(3,e_password);

			SHA hash=new SHA();
			String hash_password=hash.doEncryption(d_password);

			Connection con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/user","root","");

			Statement st= con.createStatement();

			String sql="select hash_code from user_details where user_id='"+uname+"';";

			ResultSet rs=st.executeQuery(sql);
			rs.next();
			String db_hash_code=rs.getString(1);
			con.close();
			DataOutputStream dos=new DataOutputStream(s.getOutputStream());
			if(hash_password.equals(db_hash_code))
			{
				dos.writeUTF("logged in successfully!");
			}
			else
			{
				dos.writeUTF("wrong password!!");
			}
		}
}
class conversion
```

```
{
	char temp,t;
	int asci=0;
	public String convert(int key,String inputmsg)
	{
		String op="";
		int min=0,max=0,flag=0;
		for(int i=0;i<inputmsg.length();i++)
		{
			temp=inputmsg.charAt(i);
			asci=temp+key;
			if (temp>=97&&temp<=122)
			{

				min=97;
				max=122;
				flag=1;
			}
			else if(temp>=48&&temp<=57)
			{
				min=48;
				max=57;
				flag=1;
			}
			else if(temp>=65&&temp<=90)
			{
				min=65;
				max=90;
				flag=1;
			}
			else
				flag=0;

			if(flag==1)
			{
				if(asci>max)
				{
					int rem=asci-max;
					t=(char)((min-1)+rem);
				}
				else if(asci<min)
				{
					int rem=min-asci;
					t=(char)((max+1)-rem);
				}
				else
				{
					t=(char)(asci);
				}
				op=op+t;
			}
			else
```

```
                    {
                            op=op+temp;
                    }
            }
            return op;
    }
}

class SHA{
    public String doEncryption(String text) throws Exception{
            MessageDigest md=MessageDigest.getInstance("SHA-1");
            byte[] msg=md.digest(text.getBytes());
            BigInteger bigInt=new BigInteger(1,msg);
            String hashValue=bigInt.toString(16);
            while(hashValue.length()<32)
                    hashValue+=0+hashValue;
            return hashValue;
    }
}
```

---

## Firewall-2

---

prg3_client.java

```
import java.io.*;
import java.util.*;
import java.net.*;
class prg3_client
{
    public static void main(String args[])throws Exception
    {
            Scanner sc=new Scanner(System.in);
            System.out.print("Enter packet you want to send to server:");
            String packet=sc.nextLine();
            Socket s=new Socket("127.0.0.1",1234);
            DataOutputStream dos=new DataOutputStream(s.getOutputStream());
            dos.writeUTF(packet);
            DataInputStream dis=new DataInputStream(s.getInputStream());
            String server_msg=dis.readUTF();
            System.out.println(server_msg);
            s.close();

    }
}
```

prg3_firewall.java

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

```
}
import java.net.*;
class prg3_firewall
{
	public static void main(String args[])throws Exception
	{
		ServerSocket ss=new ServerSocket(1234);
		Socket s1=ss.accept();
		DataInputStream dis=new DataInputStream(s1.getInputStream());
		String client_msg=dis.readUTF();
		String chk_pck=client_msg.toLowerCase();
		String threat="terrorist";
		StringTokenizer st=new StringTokenizer(chk_pck," ");
		String err="";
		int flag=0;
		DataOutputStream dos=new DataOutputStream(s1.getOutputStream());
		while(st.hasMoreTokens())
		{
			if(threat.equals(st.nextToken()))
			{

				err="Threat in package.. can't be delivered ";
				dos.writeUTF(err);
				s1.close();
				flag=1;
				break;
			}
		}
		if(flag==0)
		{
			Socket s2=new Socket("127.0.0.1",5678);
			DataOutputStream dos1=new DataOutputStream(s2.getOutputStream());
			dos1.writeUTF(client_msg);
			DataInputStream dis1=new DataInputStream(s2.getInputStream());
			String ack=dis1.readUTF();
			if(ack.equals("1"))
			{
				dos.writeUTF("packet recieved");
			}
			else
			{
				dos.writeUTF("unable to reach server!!");
			}
			s1.close();
			s2.close();
		}
		ss.close();
	}
}
```

prg3_server.java

```
import java.io.*;
import java.util.*;
import java.net.*;
class prg3_server
{
	public static void main(String args[])throws Exception
	{
		ServerSocket ss=new ServerSocket(5678);
		Socket s=ss.accept();
		DataInputStream dis=new DataInputStream(s.getInputStream());
		String client_msg=dis.readUTF();
		System.out.println("client packet:"+client_msg);
		DataOutputStream dos=new DataOutputStream(s.getOutputStream());
		dos.writeUTF("1");
		s.close();
		ss.close();
	}
}
```

---

## NAT

---

prg4_client.java

```
import java.io.*;
import java.util.*;
import java.net.*;
class prg4_client
{
	public static void main(String args[])throws Exception
	{
		Scanner sc=new Scanner(System.in);
		String numbers="";
		System.out.print("Enter set of numbers you want to send :");
		String num=sc.nextLine();
		Socket s=new Socket("127.0.0.1",1234);
		DataOutputStream dos=new DataOutputStream(s.getOutputStream());
		dos.writeUTF(num);
		DataInputStream dis=new DataInputStream(s.getInputStream());
		String server_msg=dis.readUTF();
		System.out.println(server_msg);
		s.close();

	}
}
```

prg4_forwarder.java

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

```
class prg4_forwarder
{
        public static void main(String args[])throws Exception
        {
                ServerSocket ss=new ServerSocket(1234);
                Socket s=ss.accept();
                DataInputStream dis=new DataInputStream(s.getInputStream());
                DataOutputStream dos=new DataOutputStream(s.getOutputStream());
                String client_msg=dis.readUTF();
                String n=client_msg.toLowerCase();
                StringTokenizer st=new StringTokenizer(n," ");
                int flag1=0,flag2=0,count=0;
                Socket s1=new Socket("127.0.0.1",5678);
                Socket s2=new Socket("127.0.0.1",5679);
                DataOutputStream dos1=new DataOutputStream(s1.getOutputStream());
                DataInputStream dis1=new DataInputStream(s1.getInputStream());
                DataOutputStream dos2=new DataOutputStream(s2.getOutputStream());
                DataInputStream dis2=new DataInputStream(s2.getInputStream());
                String server1_msg="",server2_msg="";
                while(st.hasMoreTokens())
                {
                        int num=Integer.parseInt(st.nextToken());
                        if(num%2==0)
                        {
                                server2_msg=server2_msg+" "+num;
                        }
                        else
                        {
                                server1_msg=server1_msg+" "+num;
                        }
                }
                dos1.writeUTF(server1_msg);
                dos2.writeUTF(server2_msg);
                String ack1=dis1.readUTF();
                String ack2=dis2.readUTF();
                if(ack1.equals("1")&&ack2.equals("1"))
                {
                        dos.writeUTF("packets delivered to servers");
                }
                else
                {
                        dos.writeUTF("packets not delivered to servers");
                }
                ss.close();
                s.close();
                s1.close();
                s2.close();
        }
}
```

prg4_server1.java

```
import java.io.*;
import java.util.*;
import java.net.*;
class prg4_server1
{
        public static void main(String args[])throws Exception
        {
                ServerSocket ss=new ServerSocket(5678);
                Socket s=ss.accept();
                DataInputStream dis=new DataInputStream(s.getInputStream());
                                DataOutputStream dos=new DataOutputStream(s.getOutputStream());
                String client_msg=dis.readUTF();
                if(client_msg.equals(""))
                {
                        dos.writeUTF("0");
                }
                else
                {
                System.out.println("client packet:"+client_msg);

                dos.writeUTF("1");
                }
                s.close();
                ss.close();
        }
}
```

prg4_server2.java

```
import java.io.*;
import java.util.*;
import java.net.*;
class prg4_server2
{
        public static void main(String args[])throws Exception
        {
                ServerSocket ss=new ServerSocket(5679);
                Socket s=ss.accept();
                DataInputStream dis=new DataInputStream(s.getInputStream());
                DataOutputStream dos=new DataOutputStream(s.getOutputStream());
                String client_msg=dis.readUTF();
                if(client_msg.equals(""))
                {
                        dos.writeUTF("0");
                }
                else
                {
                System.out.println("client packet:"+client_msg);

                dos.writeUTF("1");
                }
                s.close();
                ss.close();
        }
}
```

## Key Distribution

prg5_client.java

```
package dissertation;
import java.io.DataInputStream;
import java.net.Socket;
import javax.crypto.Cipher;
import javax.crypto.spec.SecretKeySpec;

public class prg5_client {
    static String receiverid;
        static SecretKeySpec receiverkey;
        //static byte[] encryptedreceiverid,encryptedsenderid,encryptedsessionkeyclient;

    public static void main(String[] args) throws Exception{
        System.out.println("client");
      receiverid="receiver123";
      receiverkey=new SecretKeySpec("12345678".getBytes(),"DES");

      Socket s=new Socket("localhost",9090);
        DataInputStream dis=new DataInputStream(s.getInputStream());

        byte[] encryptedsenderid=new byte[dis.readInt()];
        dis.readFully(encryptedsenderid);

        byte[] encryptedreceiverid=new byte[dis.readInt()];
        dis.readFully(encryptedreceiverid);

        byte[] encryptedsessionkeyclient=new byte[dis.readInt()];
        dis.readFully(encryptedsessionkeyclient);

        Cipher cipher=Cipher.getInstance("DES");
        cipher.init(Cipher.DECRYPT_MODE,receiverkey);
        byte[] senderid=cipher.doFinal(encryptedsenderid);
        System.out.println("sender id" +new String(senderid));
        byte[] receiverid=cipher.doFinal(encryptedreceiverid);
        System.out.println("receiverid" +new String(receiverid));
        byte[] sessionkey=cipher.doFinal(encryptedsessionkeyclient);
        System.out.println("sessionkey" + new String(sessionkey));
    }
}
```

prg5_KDC.java

```
package dissertation;

import java.io.DataOutputStream;
import java.net.ServerSocket;
import java.net.Socket;
import java.security.SecureRandom;
import javax.crypto.Cipher;
import javax.crypto.spec.SecretKeySpec;

public class prg5_KDC {
    static SecretKeySpec senderkey,receiverkey;
      static  byte [] sessionkey,encryptedsessionkey;
       static  String senderid,receiverid;
    public static void main(String[] args) throws Exception {
        System.out.println("KDC");
        receiverid="receiver123";
        senderid="sender123";
        receiverkey=new SecretKeySpec("12345678".getBytes(),"DES");
        senderkey=new SecretKeySpec("87654321".getBytes(),"DES");
        ServerSocket ss=new ServerSocket(8080);
              Socket s=ss.accept();
          sessionkey=generateSessionKey();
          System.out.println("sessionkey" +new String(sessionkey));
          DataOutputStream dos=new DataOutputStream(s.getOutputStream());
          Cipher cipher=Cipher.getInstance("DES");
            cipher.init(Cipher.ENCRYPT_MODE,senderkey);
            encryptedsessionkey=cipher.doFinal(sessionkey);
            cipher.init(Cipher.ENCRYPT_MODE,receiverkey);
            byte[] encryptedreceiverid=cipher.doFinal(receiverid.getBytes());
            byte[] encryptedsenderid=cipher.doFinal(senderid.getBytes());
            byte[] encryptedsessionkeyclient=cipher.doFinal(sessionkey);

                        dos.writeInt(encryptedsessionkey.length);
                        dos.write(encryptedsessionkey,0,encryptedsessionkey.length);

                        dos.writeInt(encryptedsenderid.length);
                        dos.write(encryptedsenderid,0,encryptedsenderid.length);

                        dos.writeInt(encryptedreceiverid.length);
                        dos.write(encryptedreceiverid,0,encryptedreceiverid.length);

                        dos.writeInt(encryptedsessionkeyclient.length);
                        dos.write(encryptedsessionkeyclient,0,encryptedsessionkeyclient.length);


    }
    public static byte [] generateSessionKey() throws Exception
            {
                        sessionkey=new byte[8];
                        SecureRandom random = new SecureRandom();
                        random.nextBytes(sessionkey);
```

```
                return sessionkey;
        }
}
```

prg5_server.java

```
package dissertation;
import java.io.DataOutputStream;
import java.io.DataInputStream;
import java.net.ServerSocket;
import java.net.Socket;
import javax.crypto.Cipher;
import javax.crypto.spec.SecretKeySpec;

public class prg5_server {
    static String senderid;
        static SecretKeySpec senderkey;
        static byte[] encryptedreceiverid,encryptedsenderid,encryptedsessionkeyclient;
   public static void main(String[] args) throws Exception{
       System.out.println("Server");
     senderid="sender123";
     senderkey=new SecretKeySpec("87654321".getBytes(),"DES");
     getSessionInfoServer();
     ServerSocket ss=new ServerSocket(9090);
        Socket s=ss.accept();
        DataOutputStream dos=new DataOutputStream(s.getOutputStream());
        dos.writeInt(encryptedsenderid.length);
        dos.write(encryptedsenderid,0,encryptedsenderid.length);

        dos.writeInt(encryptedreceiverid.length);
        dos.write(encryptedreceiverid,0,encryptedreceiverid.length);

        dos.writeInt(encryptedsessionkeyclient.length);
        dos.write(encryptedsessionkeyclient,0,encryptedsessionkeyclient.length);

   }
   public static void getSessionInfoServer() throws Exception
        {
                Socket s=new Socket("localhost",8080);
                DataInputStream dis=new DataInputStream(s.getInputStream());

                byte[] encryptedsessionkey=new byte[dis.readInt()];
           dis.readFully(encryptedsessionkey);

                encryptedsenderid=new byte[dis.readInt()];
           dis.readFully(encryptedsenderid);

                encryptedreceiverid=new byte[dis.readInt()];
                dis.readFully(encryptedreceiverid);
```

```
              encryptedsessionkeyclient=new byte[dis.readInt()];
        dis.readFully(encryptedsessionkeyclient);

              Cipher cipher=Cipher.getInstance("DES");
              cipher.init(Cipher.DECRYPT_MODE,senderkey);
              byte[] sessionkey=cipher.doFinal(encryptedsessionkey);
              System.out.println("serversessionkey" +new String(sessionkey));
         }
}
```