Basic Implementation:

AES:

```
package javae;

import java.io.UnsupportedEncodingException;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.util.Arrays;
import java.util.Base64;

import javax.crypto.Cipher;
import javax.crypto.spec.SecretKeySpec;

public class AES {

    private static SecretKeySpec secretKey;
    private static byte[] key;

    public static void setKey(String myKey) {
        MessageDigest sha = null;
        try {
            key = myKey.getBytes("UTF-8");
            sha = MessageDigest.getInstance("SHA-1");
            key = sha.digest(key);
            key = Arrays.copyOf(key, 16);
            secretKey = new SecretKeySpec(key, "AES");
        } catch (NoSuchAlgorithmException e) {
            e.printStackTrace();
        } catch (UnsupportedEncodingException e) {
            e.printStackTrace();
        }
    }

    public static String encrypt(String strToEncrypt, String secret) {
        try {
            setKey(secret);
            Cipher cipher = Cipher.getInstance("AES/ECB/PKCS5Padding");
            cipher.init(Cipher.ENCRYPT_MODE, secretKey);
```

```
            return
Base64.getEncoder().encodeToString(cipher.doFinal(strToEncrypt.getBytes("UTF-8")));
        } catch (Exception e) {
            System.out.println("Error while encrypting: " + e.toString());
        }
        return null;
    }

    public static String decrypt(String strToDecrypt, String secret) {
        try {
            setKey(secret);
            Cipher cipher = Cipher.getInstance("AES/ECB/PKCS5PADDING");
            cipher.init(Cipher.DECRYPT_MODE, secretKey);
            return new String(cipher.doFinal(Base64.getDecoder().decode(strToDecrypt)));
        } catch (Exception e) {
            System.out.println("Error while decrypting: " + e.toString());
        }
        return null;
    }

    public static void main(String[] args) {
        final String secretKey = "hello there   ";
        String originalString  = "general kenobi";
        String encryptedString = AES.encrypt(originalString, secretKey);
        String decryptedString = AES.decrypt(encryptedString, secretKey);

        System.out.println("Original  - " + originalString);
        System.out.println("Encrypted - " + encryptedString);
        System.out.println("Decrypted - " + decryptedString);
    }
}
```

Output:

```
shankar@shankar-ThinkPad-L450:~/Documents/AU/sem6/security/lab/week6/javae$ java AES.java
Original  - general kenobi
Encrypted - Ykb/PxADXfwGONgxocY4cQ==
Decrypted - general kenobi
```

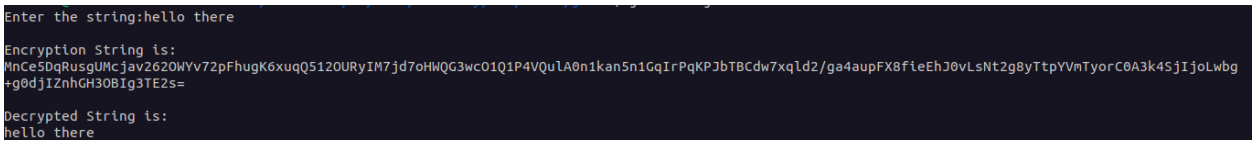
RSA:

```
package javae;

import javax.crypto.BadPaddingException;
import javax.crypto.Cipher;
import javax.crypto.IllegalBlockSizeException;
import javax.crypto.NoSuchPaddingException;

import java.io.*;
import java.security.*;
import java.security.spec.InvalidKeySpecException;
import java.security.spec.PKCS8EncodedKeySpec;
import java.security.spec.X509EncodedKeySpec;
import java.util.Base64;

public class RSA {

    private static String publicKey =
"MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCgFGVfrY4jQSoZQWWygZ83roKXWD
4YeT2x2p41dGkPixe73rT2IW04glagN2vgoZoHuOPqa5and6kAmK2ujmCHu6D1auJhE2tXP+yL
kpSiYMQucDKmCsWMnW9XlC5K7OSL77TXXcfvTvyZcjObEz6LIBRzs6+FqpFbUO9SJEfh6wID
AQAB";
    private static String privateKey =
"MIICdQIBADANBgkqhkiG9w0BAQEFAASCAl8wggJbAgEAAoGBAKAUZV+tjiNBKhlBZbKBnze
ugpdYPhh5PbHanjV0aQ+LF7vetPYhbTiCVqA3a+Chmge44+prlqd3qQCYra6OYIe7oPVq4mETa
1c/7IuSlKJgxC5wMqYKxYydb1eULkrs5IvvtNddx+9O/JlyM5sTPosgFHOzr4WqkVtQ71lkR+HrAg
MBAAECgYAkQLo8kteP0GAyXAcmCAkA2Tql/8wASuTX9ITD4lsws/VqDKO64hMUKyBnJGX/9
1kkypCDNF5oCsdxZSJgV8owViYWZPnbvEcNqLtqgs7nj1UHuX9S5yYIPGN/mHL6OJJ7sosOd6
rqdpg6JRRkAKUV+tmN/7Gh0+GFXM+ug6mgwQJBAO9/+CWpCAVoGxCA+YsTMb82fTOmGY
MkZOAfQsvIV2v6DC8eJrSa+c0yCOTa3tirlCkhBfB08f8U2iEPS+Gu3bECQQCrG7O0gYmFL2R
X1O+37ovyyHTbst4s4xbLW4jLzbSoimL235lCdIC+fllEEP96wPAiqo6dzmdH8KsGmVozsVRbAk
B0ME8AZjp/9Pt8TDXD5LHzo8mlruUdnCBcIo5TMoRG2+3hRe1dHPonNCjgbdZCoyqjsWOiPfn
Q2Brigvs7J4xhAkBGRiZUKC92x7QKbqXVgN9xYuq7oIanIM0nz/wq190uq0dh5Qtow7hshC/dSK
3kmIEHe8z++tpoLWvQVgM538apAkBoSNfaTkDZhFavuiVl6L8cWCoDcJBItip8wKQhXwHp0O3
HLg10OEd14M58ooNfpgt+8D8/8/2OOFaR0HzA+2Dm";

    public static PublicKey getPublicKey(String base64PublicKey) {
        PublicKey publicKey = null;
        try {
```

```
            X509EncodedKeySpec keySpec = new
X509EncodedKeySpec(Base64.getDecoder().decode(base64PublicKey.getBytes()));
            KeyFactory keyFactory = KeyFactory.getInstance("RSA");
            publicKey = keyFactory.generatePublic(keySpec);
            return publicKey;
        } catch (NoSuchAlgorithmException e) {
            e.printStackTrace();
        } catch (InvalidKeySpecException e) {
            e.printStackTrace();
        }
        return publicKey;
    }

    public static PrivateKey getPrivateKey(String base64PrivateKey) {
        PrivateKey privateKey = null;
        PKCS8EncodedKeySpec keySpec = new
PKCS8EncodedKeySpec(Base64.getDecoder().decode(base64PrivateKey.getBytes()));
        KeyFactory keyFactory = null;
        try {
            keyFactory = KeyFactory.getInstance("RSA");
        } catch (NoSuchAlgorithmException e) {
            e.printStackTrace();
        }
        try {
            privateKey = keyFactory.generatePrivate(keySpec);
        } catch (InvalidKeySpecException e) {
            e.printStackTrace();
        }
        return privateKey;
    }

    public static byte[] encrypt(String data, String publicKey) throws BadPaddingException,
IllegalBlockSizeException,
            InvalidKeyException, NoSuchPaddingException, NoSuchAlgorithmException {
        Cipher cipher = Cipher.getInstance("RSA/ECB/PKCS1Padding");
        cipher.init(Cipher.ENCRYPT_MODE, getPublicKey(publicKey));
        return cipher.doFinal(data.getBytes());
    }

    public static String decrypt(byte[] data, PrivateKey privateKey) throws
NoSuchPaddingException,
            NoSuchAlgorithmException, InvalidKeyException, BadPaddingException,
IllegalBlockSizeException {
        Cipher cipher = Cipher.getInstance("RSA/ECB/PKCS1Padding");
```

```
        cipher.init(Cipher.DECRYPT_MODE, privateKey);
        return new String(cipher.doFinal(data));
    }

    public static String decrypt(String data, String base64PrivateKey) throws
IllegalBlockSizeException,
            InvalidKeyException, BadPaddingException, NoSuchAlgorithmException,
NoSuchPaddingException {
        return decrypt(Base64.getDecoder().decode(data.getBytes()),
getPrivateKey(base64PrivateKey));
    }

    public static void main(String[] args) throws IllegalBlockSizeException, InvalidKeyException,
            NoSuchPaddingException, BadPaddingException, IOException {
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        try {

            System.out.print("Enter the string:");
            String inputString = br.readLine();

            String encryptedString = Base64.getEncoder().encodeToString(encrypt(inputString,
publicKey));
            System.out.println("\nEncryption String is:\n" + encryptedString);

            String decryptedString = RSA.decrypt(encryptedString, privateKey);
            System.out.println("\nDecrypted String is:\n" + decryptedString);
        } catch (NoSuchAlgorithmException e) {
            System.err.println(e.getMessage());
        }

    }
}
```

Output:



Files- RSA

```
import javax.crypto.BadPaddingException;
import javax.crypto.Cipher;
import javax.crypto.IllegalBlockSizeException;
import javax.crypto.NoSuchPaddingException;

import java.io.*;
import java.security.*;
import java.security.spec.InvalidKeySpecException;
import java.security.spec.PKCS8EncodedKeySpec;
import java.security.spec.X509EncodedKeySpec;
import java.util.Base64;
import java.util.Scanner;

package RSA;

import javax.crypto.BadPaddingException;
import javax.crypto.Cipher;
import javax.crypto.IllegalBlockSizeException;
import javax.crypto.NoSuchPaddingException;

import java.io.*;
import java.security.*;
import java.security.spec.InvalidKeySpecException;
import java.security.spec.PKCS8EncodedKeySpec;
import java.security.spec.X509EncodedKeySpec;
import java.util.Base64;
import java.util.Scanner;

public class RSA {

    private static String publicKey =
"MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCgFGVfrY4jQSoZQWWygZ83roKXWD
4YeT2x2p41dGkPixe73rT2IW04glagN2vgoZoHuOPqa5and6kAmK2ujmCHu6D1auJhE2tXP+yL
```

```
kpSiYMQucDKmCsWMnW9XlC5K7OSL77TXXcfvTvyZcjObEz6LIBRzs6+FqpFbUO9SJEfh6wID
AQAB";
    private static String privateKey =
"MIICdQIBADANBgkqhkiG9w0BAQEFAASCAl8wggJbAgEAAoGBAKAUZV+tjiNBKhlBZbKBnze
ugpdYPhh5PbHanjV0aQ+LF7vetPYhbTiCVqA3a+Chmge44+prlqd3qQCYra6OYIe7oPVq4mETa
1c/7IuSlKJgxC5wMqYKxYydb1eULkrs5IvvtNddx+9O/JlyM5sTPosgFHOzr4WqkVtQ71lkR+HrAg
MBAAECgYAkQLo8kteP0GAyXAcmCAkA2Tql/8wASuTX9ITD4lsws/VqDKO64hMUKyBnJGX/9
1kkypCDNF5oCsdxZSJgV8owViYWZPnbvEcNqLtqgs7nj1UHuX9S5yYIPGN/mHL6OJJ7sosOd6
rqdpg6JRRkAKUV+tmN/7Gh0+GFXM+ug6mgwQJBAO9/+CWpCAVoGxCA+YsTMb82fTOmGY
MkZOAfQsvIV2v6DC8eJrSa+c0yCOTa3tirlCkhBfB08f8U2iEPS+Gu3bECQQCrG7O0gYmFL2R
X1O+37ovyyHTbst4s4xbLW4jLzbSoimL235lCdIC+fllEEP96wPAiqo6dzmdH8KsGmVozsVRbAk
B0ME8AZjp/9Pt8TDXD5LHzo8mlruUdnCBcIo5TMoRG2+3hRe1dHPonNCjgbdZCoyqjsWOiPfn
Q2Brigvs7J4xhAkBGRiZUKC92x7QKbqXVgN9xYuq7oIanIM0nz/wq190uq0dh5Qtow7hshC/dSK
3kmIEHe8z++tpoLWvQVgM538apAkBoSNfaTkDZhFavuiVl6L8cWCoDcJBItip8wKQhXwHp0O3
HLg10OEd14M58ooNfpgt+8D8/8/2OOFaR0HzA+2Dm";

   public static PublicKey getPublicKey(String base64PublicKey) {
      PublicKey publicKey = null;
      try {
         X509EncodedKeySpec keySpec = new
X509EncodedKeySpec(Base64.getDecoder().decode(base64PublicKey.getBytes()));
         KeyFactory keyFactory = KeyFactory.getInstance("RSA");
         publicKey = keyFactory.generatePublic(keySpec);
         return publicKey;
      } catch (NoSuchAlgorithmException e) {
         e.printStackTrace();
      } catch (InvalidKeySpecException e) {
         e.printStackTrace();
      }
      return publicKey;
   }

   public static PrivateKey getPrivateKey(String base64PrivateKey) {
      PrivateKey privateKey = null;
      PKCS8EncodedKeySpec keySpec = new
PKCS8EncodedKeySpec(Base64.getDecoder().decode(base64PrivateKey.getBytes()));
      KeyFactory keyFactory = null;
      try {
         keyFactory = KeyFactory.getInstance("RSA");
      } catch (NoSuchAlgorithmException e) {
         e.printStackTrace();
      }
      try {
         privateKey = keyFactory.generatePrivate(keySpec);
```

```
        } catch (InvalidKeySpecException e) {
            e.printStackTrace();
        }
        return privateKey;
    }

    public static byte[] encrypt(String data, String publicKey) throws BadPaddingException,
IllegalBlockSizeException,
            InvalidKeyException, NoSuchPaddingException, NoSuchAlgorithmException {
        Cipher cipher = Cipher.getInstance("RSA/ECB/PKCS1Padding");
        cipher.init(Cipher.ENCRYPT_MODE, getPublicKey(publicKey));
        return cipher.doFinal(data.getBytes());
    }

    public static String decrypt(byte[] data, PrivateKey privateKey) throws
NoSuchPaddingException,
            NoSuchAlgorithmException, InvalidKeyException, BadPaddingException,
IllegalBlockSizeException {
        Cipher cipher = Cipher.getInstance("RSA/ECB/PKCS1Padding");
        cipher.init(Cipher.DECRYPT_MODE, privateKey);
        return new String(cipher.doFinal(data));
    }

    public static String decrypt(String data, String base64PrivateKey) throws
IllegalBlockSizeException,
            InvalidKeyException, BadPaddingException, NoSuchAlgorithmException,
NoSuchPaddingException {
        return decrypt(Base64.getDecoder().decode(data.getBytes()),
getPrivateKey(base64PrivateKey));
    }

    public static void main(String[] args) throws IllegalBlockSizeException, InvalidKeyException,
            NoSuchPaddingException, BadPaddingException, IOException {
        String input = null;
        String encryptedString = null;
        String decryptedString = null;

        try {
            File myObj = new File("txts/inputRSA.txt");
            Scanner sc = new Scanner(myObj);
            while (sc.hasNextLine()) {
                input = sc.nextLine();
            }
            sc.close();
```

```
    } catch (FileNotFoundException e) {
    }

    try {
        File myObj = new File("txts/encryptRSA.txt");
        if (myObj.createNewFile()) {

        }
    } catch (IOException e) {
    }

    try {
        File myObj = new File("txts/decryptRSA.txt");
        if (myObj.createNewFile()) {

        }
    } catch (IOException e) {
    }

    try {
        encryptedString = Base64.getEncoder().encodeToString(encrypt(input, publicKey));
    } catch (NoSuchAlgorithmException e) {
    }

    try {
        FileWriter myWriter = new FileWriter("txts/encryptRSA.txt");
        myWriter.write("\n" + encryptedString + "\n");
        myWriter.close();
    } catch (IOException e) {
    }

    try {
        decryptedString = RSA.decrypt(encryptedString, privateKey);
    } catch (NoSuchAlgorithmException e) {
    }

    try {
        FileWriter myWriter = new FileWriter("txts/decryptRSA.txt");
        myWriter.write("\n" + decryptedString + "\n");
        myWriter.close();
    } catch (IOException e) {
    }

}
```

}

Output:

```
shankar@shankar-ThinkPad-L450:~/Documents/AU/sem6/security/lab/week7/RSA$ ls txts/
inputRSA.txt
shankar@shankar-ThinkPad-L450:~/Documents/AU/sem6/security/lab/week7/RSA$ cat txts/inputRSA.txt
Shankar Subramanianshankar@shankar-ThinkPad-L450:~/Documents/AU/sem6/security/lab/week7/RSA$
shankar@shankar-ThinkPad-L450:~/Documents/AU/sem6/security/lab/week7/RSA$ java RSA.java
shankar@shankar-ThinkPad-L450:~/Documents/AU/sem6/security/lab/week7/RSA$ ls txts/
decryptRSA.txt  encryptRSA.txt  inputRSA.txt
```

```
shankar@shankar-ThinkPad-L450:~/Documents/AU/sem6/security/lab/week7/RSA/txts$ ls
decryptRSA.txt  encryptRSA.txt  inputRSA.txt
shankar@shankar-ThinkPad-L450:~/Documents/AU/sem6/security/lab/week7/RSA/txts$ cat *

Shankar Subramanian

Lz3EbGtS/ER9zx15IFjUB2EAUCck7z6S2qqVwBH+2jqUQhqgd9yJ2kX4t5m9IbO/7sH3BKW7fwuitdi2MsHXmxa7AepbUwtXk4H1oNIDfWsQyo05x9tzPTRv3YjYkMD0Xjv+1h24Ynfa8CHLJWHt5h
P04JzdoLg53503A6O9kKA=
Shankar Subramanianshankar@shankar-ThinkPad-L450:~/Documents/AU/sem6/security/lab/week7/RSA/txts$
```

Files- AES

```
import javax.crypto.BadPaddingException;
import javax.crypto.Cipher;
import javax.crypto.IllegalBlockSizeException;
import javax.crypto.NoSuchPaddingException;

import java.io.*;
import java.security.*;
import java.security.spec.InvalidKeySpecException;
import java.security.spec.PKCS8EncodedKeySpec;
import java.security.spec.X509EncodedKeySpec;
import java.util.Base64;
import java.util.Scanner;

package RSA;

import javax.crypto.BadPaddingException;
import javax.crypto.Cipher;
import javax.crypto.IllegalBlockSizeException;
import javax.crypto.NoSuchPaddingException;
```

```
import java.io.*;
import java.security.*;
import java.security.spec.InvalidKeySpecException;
import java.security.spec.PKCS8EncodedKeySpec;
import java.security.spec.X509EncodedKeySpec;
import java.util.Base64;
import java.util.Scanner;

public class RSA {

 private static SecretKeySpec secretKey;
   private static byte[] key;

   public static void setKey(String myKey) {
      MessageDigest sha = null;
      try {
         key = myKey.getBytes("UTF-8");
         sha = MessageDigest.getInstance("SHA-1");
         key = sha.digest(key);
         key = Arrays.copyOf(key, 16);
         secretKey = new SecretKeySpec(key, "AES");
      } catch (NoSuchAlgorithmException e) {
         e.printStackTrace();
      } catch (UnsupportedEncodingException e) {
         e.printStackTrace();
      }
   }

   public static String encrypt(String strToEncrypt, String secret) {
      try {
         setKey(secret);
         Cipher cipher = Cipher.getInstance("AES/ECB/PKCS5Padding");
         cipher.init(Cipher.ENCRYPT_MODE, secretKey);
         return
Base64.getEncoder().encodeToString(cipher.doFinal(strToEncrypt.getBytes("UTF-8")));
      } catch (Exception e) {
         System.out.println("Error while encrypting: " + e.toString());
      }
      return null;
   }

   public static String decrypt(String strToDecrypt, String secret) {
      try {
         setKey(secret);
```

```
            Cipher cipher = Cipher.getInstance("AES/ECB/PKCS5PADDING");
            cipher.init(Cipher.DECRYPT_MODE, secretKey);
            return new String(cipher.doFinal(Base64.getDecoder().decode(strToDecrypt)));
        } catch (Exception e) {
            System.out.println("Error while decrypting: " + e.toString());
        }
        return null;
    }


    public static void main(String[] args) throws IllegalBlockSizeException, InvalidKeyException,
            NoSuchPaddingException, BadPaddingException, IOException {
        String input = null;
        String encryptedString = null;
        String decryptedString = null;

        try {
            File myObj = new File("txts/inputRSA.txt");
            Scanner sc = new Scanner(myObj);
            while (sc.hasNextLine()) {
                input = sc.nextLine();
            }
            sc.close();
        } catch (FileNotFoundException e) {
        }

        try {
            File myObj = new File("txts/encryptRSA.txt");
            if (myObj.createNewFile()) {

            }
        } catch (IOException e) {
        }

        try {
            File myObj = new File("txts/decryptRSA.txt");
            if (myObj.createNewFile()) {

            }
        } catch (IOException e) {
        }

        try {
            encryptedString = Base64.getEncoder().encodeToString(encrypt(input, publicKey));
```

```
        } catch (NoSuchAlgorithmException e) {
        }

        try {
            FileWriter myWriter = new FileWriter("txts/encryptRSA.txt");
            myWriter.write("\n" + encryptedString + "\n");
            myWriter.close();
        } catch (IOException e) {
        }

        try {
            decryptedString = RSA.decrypt(encryptedString, privateKey);
        } catch (NoSuchAlgorithmException e) {
        }

        try {
            FileWriter myWriter = new FileWriter("txts/decryptRSA.txt");
            myWriter.write("\n" + decryptedString + "\n");
            myWriter.close();
        } catch (IOException e) {
        }

    }
}
```

Output:

```
shankar@shankar-ThinkPad-L450:~/Documents/AU/sem6/security/lab/week7/RSA$ ls txts/
inputRSA.txt
shankar@shankar-ThinkPad-L450:~/Documents/AU/sem6/security/lab/week7/RSA$ cat txts/inputRSA.txt
Shankar Subramanianshankar@shankar-ThinkPad-L450:~/Documents/AU/sem6/security/lab/week7/RSA$
shankar@shankar-ThinkPad-L450:~/Documents/AU/sem6/security/lab/week7/RSA$ java RSA.java
shankar@shankar-ThinkPad-L450:~/Documents/AU/sem6/security/lab/week7/RSA$ ls txts/
decryptRSA.txt  encryptRSA.txt  inputRSA.txt
```

```
shankar@shankar-ThinkPad-L450:~/Documents/AU/sem6/security/lab/week7/RSA/txts$ ls
decryptRSA.txt  encryptRSA.txt  inputRSA.txt
shankar@shankar-ThinkPad-L450:~/Documents/AU/sem6/security/lab/week7/RSA/txts$ cat *

Shankar Subramanian

Lz3EbGtS/ER9zx15IFjUB2EAUCck7z6S2qqVwBH+2jqUQhqgd9yJ2kX4t5m9IbO/7sH3BKW7fwuitdi2MsHXmxa7AepbUwtXk4H1oNIDfWsQyo05x9tzPTRv3YjYkMD0Xjv+1h24Ynfa8CHLJWHt5hP04JzdoLg53503A6O9kKA=
Shankar Subramanianshankar@shankar-ThinkPad-L450:~/Documents/AU/sem6/security/lab/week7/RSA/txts$
```

Sockets- AES

Client.java

```
package AES;

import java.io.*;
import java.net.*;
import java.io.UnsupportedEncodingException;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.util.Arrays;
import java.util.Base64;
import java.util.Scanner;

import javax.crypto.Cipher;
import javax.crypto.spec.SecretKeySpec;

public class Client {

    private static SecretKeySpec secretKey;
    private static byte[] key;

    public static void setKey(String myKey) {
        MessageDigest sha = null;
        try {
            key = myKey.getBytes("UTF-8");
            sha = MessageDigest.getInstance("SHA-1");
            key = sha.digest(key);
            key = Arrays.copyOf(key, 16);
            secretKey = new SecretKeySpec(key, "AES");
        } catch (NoSuchAlgorithmException e) {
            e.printStackTrace();
        } catch (UnsupportedEncodingException e) {
            e.printStackTrace();
        }
    }
```

```
    public static String encrypt(String strToEncrypt, String secret) {
        try {
            setKey(secret);
            Cipher cipher = Cipher.getInstance("AES/ECB/PKCS5Padding");
            cipher.init(Cipher.ENCRYPT_MODE, secretKey);
            return
Base64.getEncoder().encodeToString(cipher.doFinal(strToEncrypt.getBytes("UTF-8")));
        } catch (Exception e) {
            System.out.println("Error while encrypting: " + e.toString());
        }
        return null;
    }

    public static void main(String[] args) {
        try {
            Socket s = new Socket("localhost", 6666);
            DataOutputStream out = new DataOutputStream(s.getOutputStream());
            Scanner sc = new Scanner(System.in);
            String secretKey = "kenobi";
            System.out.print("Enter the message:");
            String originalString = sc.nextLine();

            String encryptedString = Client.encrypt(originalString, secretKey);
            out.writeUTF(encryptedString);
            out.flush();
            out.close();
            sc.close();
            s.close();
        } catch (Exception e) {
            System.out.println(e);
        }
    }
}
```

Server.java

```
package AES;

import java.io.UnsupportedEncodingException;
import java.security.MessageDigest;
```

```
import java.security.NoSuchAlgorithmException;
import java.util.Arrays;
import java.util.Base64;
import javax.crypto.Cipher;
import javax.crypto.spec.SecretKeySpec;
import java.io.*;
import java.net.*;

public class Server {

    private static SecretKeySpec secretKey;
    private static byte[] key;

    public static void setKey(String myKey) {
        MessageDigest sha = null;
        try {
            key = myKey.getBytes("UTF-8");
            sha = MessageDigest.getInstance("SHA-1");
            key = sha.digest(key);
            key = Arrays.copyOf(key, 16);
            secretKey = new SecretKeySpec(key, "AES");
        } catch (NoSuchAlgorithmException e) {
            e.printStackTrace();
        } catch (UnsupportedEncodingException e) {
            e.printStackTrace();
        }
    }

    public static String decrypt(String strToDecrypt, String secret) {
        try {
            setKey(secret);
            Cipher cipher = Cipher.getInstance("AES/ECB/PKCS5PADDING");
            cipher.init(Cipher.DECRYPT_MODE, secretKey);
            return new String(cipher.doFinal(Base64.getDecoder().decode(strToDecrypt)));
        } catch (Exception e) {
            System.out.println("Error while decrypting: " + e.toString());
        }
        return null;
    }

    public static void main(String[] args) {
        try {
            String secretKey = "kenobi";
            ServerSocket ss = new ServerSocket(6666);
```

```
            Socket s = ss.accept();// establishes connection
            DataInputStream dis = new DataInputStream(s.getInputStream());
            String encryptedString = (String) dis.readUTF();
            System.out.println("Decrypted String:" + encryptedString);
            String decrytpedString = Server.decrypt(encryptedString, secretKey);
            System.out.println("Decrypted String:" + decrytpedString);
            ss.close();
        } catch (Exception e) {
            System.out.println(e);
        }
    }
}
```

Output:

```
shankar@shankar-ThinkPad-L450:~/Documents/AU/sem6/security/lab/week7/AES$ java Client.java
Enter the message:hello there
```

```
shankar@shankar-ThinkPad-L450:~/Documents/AU/sem6/security/lab/week7/AES$ java Server.java
Decrypted String:Rd38sYNthdzmK4PECrqWAg==
Decrypted String:hello there
```

Sockets- RSA:

Client.java

```
package AES;

import javax.crypto.BadPaddingException;
import javax.crypto.Cipher;
import javax.crypto.IllegalBlockSizeException;
import javax.crypto.NoSuchPaddingException;
import java.io.*;
import java.security.*;
import java.security.spec.InvalidKeySpecException;
import java.security.spec.PKCS8EncodedKeySpec;
import java.security.spec.X509EncodedKeySpec;
import java.util.Base64;
import java.util.Scanner;
import java.io.*;
import java.net.*;
```

```
public class Client {
   private static String publicKey =
"MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCgFGVfrY4jQSoZQWWygZ83roKXWD
4YeT2x2p41dGkPixe73rT2IW04glagN2vgoZoHuOPqa5and6kAmK2ujmCHu6D1auJhE2tXP+yL
kpSiYMQucDKmCsWMnW9XlC5K7OSL77TXXcfvTvyZcjObEz6LIBRzs6+FqpFbUO9SJEfh6wID
AQAB";
   private static String privateKey =
"MIICdQIBADANBgkqhkiG9w0BAQEFAASCAl8wggJbAgEAAoGBAKAUZV+tjiNBKhlBZbKBnze
ugpdYPhh5PbHanjV0aQ+LF7vetPYhbTiCVqA3a+Chmge44+prlqd3qQCYra6OYIe7oPVq4mETa
1c/7IuSlKJgxC5wMqYKxYydb1eULkrs5IvvtNddx+9O/JlyM5sTPosgFHOzr4WqkVtQ71IkR+HrAg
MBAAECgYAkQLo8kteP0GAyXAcmCAkA2Tql/8wASuTX9ITD4lsws/VqDKO64hMUKyBnJGX/9
1kkypCDNF5oCsdxZSJgV8owViYWZPnbvEcNqLtqgs7nj1UHuX9S5yYIPGN/mHL6OJJ7sosOd6
rqdpg6JRRkAKUV+tmN/7Gh0+GFXM+ug6mgwQJBAO9/+CWpCAVoGxCA+YsTMb82fTOmGY
MkZOAfQsvIV2v6DC8eJrSa+c0yCOTa3tirlCkhBfB08f8U2iEPS+Gu3bECQQCrG7O0gYmFL2R
X1O+37ovyyHTbst4s4xbLW4jLzbSoimL235lCdIC+fllEEP96wPAiqo6dzmdH8KsGmVozsVRbAk
B0ME8AZjp/9Pt8TDXD5LHzo8mlruUdnCBcIo5TMoRG2+3hRe1dHPonNCjgbdZCoyqjsWOiPfn
Q2Brigvs7J4xhAkBGRiZUKC92x7QKbqXVgN9xYuq7oIanIM0nz/wq190uq0dh5Qtow7hshC/dSK
3kmIEHe8z++tpoLWvQVgM538apAkBoSNfaTkDZhFavuiVl6L8cWCoDcJBItip8wKQhXwHp0O3
HLg10OEd14M58ooNfpgt+8D8/8/2OOFaR0HzA+2Dm";

   public static PublicKey getPublicKey(String base64PublicKey) {
      PublicKey publicKey = null;
      try {
         X509EncodedKeySpec keySpec = new
X509EncodedKeySpec(Base64.getDecoder().decode(base64PublicKey.getBytes()));
         KeyFactory keyFactory = KeyFactory.getInstance("RSA");
         publicKey = keyFactory.generatePublic(keySpec);
         return publicKey;
      } catch (NoSuchAlgorithmException e) {
         e.printStackTrace();
      } catch (InvalidKeySpecException e) {
         e.printStackTrace();
      }
      return publicKey;
   }

   public static PrivateKey getPrivateKey(String base64PrivateKey) {
      PrivateKey privateKey = null;
      PKCS8EncodedKeySpec keySpec = new
PKCS8EncodedKeySpec(Base64.getDecoder().decode(base64PrivateKey.getBytes()));
      KeyFactory keyFactory = null;
      try {
         keyFactory = KeyFactory.getInstance("RSA");
```

```
        } catch (NoSuchAlgorithmException e) {
            e.printStackTrace();
        }
        try {
            privateKey = keyFactory.generatePrivate(keySpec);
        } catch (InvalidKeySpecException e) {
            e.printStackTrace();
        }
        return privateKey;
    }

    public static byte[] encrypt(String data, String publicKey) throws BadPaddingException,
IllegalBlockSizeException,
            InvalidKeyException, NoSuchPaddingException, NoSuchAlgorithmException {
        Cipher cipher = Cipher.getInstance("RSA/ECB/PKCS1Padding");
        cipher.init(Cipher.ENCRYPT_MODE, getPublicKey(publicKey));
        return cipher.doFinal(data.getBytes());
    }

    public static void main(String[] args) throws IllegalBlockSizeException, InvalidKeyException,
            NoSuchPaddingException, BadPaddingException, IOException {
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        try {
            System.out.print("Enter the string:");
            String inputString = br.readLine();
            String encryptedString = Base64.getEncoder().encodeToString(encrypt(inputString,
publicKey));
            System.out.println("\nEncryption String is:\n" + encryptedString);
            Socket s = new Socket("localhost", 6666);
            DataOutputStream dout = new DataOutputStream(s.getOutputStream());
            dout.writeUTF(encryptedString);
            dout.flush();
            dout.close();
            s.close();
        } catch (Exception e) {
        }
        ;
    }
}
```

Server.java

```
package AES;
```

```
import javax.crypto.BadPaddingException;
import javax.crypto.Cipher;
import javax.crypto.IllegalBlockSizeException;
import javax.crypto.NoSuchPaddingException;
import java.io.*;
import java.security.*;
import java.security.spec.InvalidKeySpecException;
import java.security.spec.PKCS8EncodedKeySpec;
import java.security.spec.X509EncodedKeySpec;
import java.util.Base64;

import java.net.*;

public class Server {
   private static String publicKey =
"MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCgFGVfrY4jQSoZQWWygZ83roKXWD
4YeT2x2p41dGkPixe73rT2IW04glagN2vgoZoHuOPqa5and6kAmK2ujmCHu6D1auJhE2tXP+yL
kpSiYMQucDKmCsWMnW9XlC5K7OSL77TXXcfvTvyZcjObEz6LIBRzs6+FqpFbUO9SJEfh6wID
AQAB";
   private static String privateKey =
"MIICdQIBADANBgkqhkiG9w0BAQEFAASCAl8wggJbAgEAAoGBAKAUZV+tjiNBKhlBZbKBnze
ugpdYPhh5PbHanjV0aQ+LF7vetPYhbTiCVqA3a+Chmge44+prlqd3qQCYra6OYIe7oPVq4mETa
1c/7IuSlKJgxC5wMqYKxYydb1eULkrs5IvvtNddx+9O/JlyM5sTPosgFHOzr4WqkVtQ71IkR+HrAg
MBAAECgYAkQLo8kteP0GAyXAcmCAkA2Tql/8wASuTX9ITD4lsws/VqDKO64hMUKyBnJGX/9
1kkypCDNF5oCsdxZSJgV8owViYWZPnbvEcNqLtqgs7nj1UHuX9S5yYIPGN/mHL6OJJ7sosOd6
rqdpg6JRRkAKUV+tmN/7Gh0+GFXM+ug6mgwQJBAO9/+CWpCAVoGxCA+YsTMb82fTOmGY
MkZOAfQsvIV2v6DC8eJrSa+c0yCOTa3tirlCkhBfB08f8U2iEPS+Gu3bECQQCrG7O0gYmFL2R
X1O+37ovyyHTbst4s4xbLW4jLzbSoimL235lCdIC+fllEEP96wPAiqo6dzmdH8KsGmVozsVRbAk
B0ME8AZjp/9Pt8TDXD5LHzo8mlruUdnCBcIo5TMoRG2+3hRe1dHPonNCjgbdZCoyqjsWOiPfn
Q2Brigvs7J4xhAkBGRiZUKC92x7QKbqXVgN9xYuq7oIanIM0nz/wq190uq0dh5Qtow7hshC/dSK
3kmIEHe8z++tpoLWvQVgM538apAkBoSNfaTkDZhFavuiVl6L8cWCoDcJBItip8wKQhXwHp0O3
HLg10OEd14M58ooNfpgt+8D8/8/2OOFaR0HzA+2Dm";

   public static PublicKey getPublicKey(String base64PublicKey) {
      PublicKey publicKey = null;
      try {
         X509EncodedKeySpec keySpec = new
X509EncodedKeySpec(Base64.getDecoder().decode(base64PublicKey.getBytes()));
         KeyFactory keyFactory = KeyFactory.getInstance("RSA");
         publicKey = keyFactory.generatePublic(keySpec);
         return publicKey;
      } catch (NoSuchAlgorithmException e) {
         e.printStackTrace();
```

```
        } catch (InvalidKeySpecException e) {
            e.printStackTrace();
        }
        return publicKey;
    }

    public static PrivateKey getPrivateKey(String base64PrivateKey) {
        PrivateKey privateKey = null;
        PKCS8EncodedKeySpec keySpec = new
PKCS8EncodedKeySpec(Base64.getDecoder().decode(base64PrivateKey.getBytes()));
        KeyFactory keyFactory = null;
        try {
            keyFactory = KeyFactory.getInstance("RSA");
        } catch (NoSuchAlgorithmException e) {
            e.printStackTrace();
        }
        try {
            privateKey = keyFactory.generatePrivate(keySpec);
        } catch (InvalidKeySpecException e) {
            e.printStackTrace();
        }
        return privateKey;
    }

    public static String decrypt(byte[] data, PrivateKey privateKey) throws
NoSuchPaddingException,
            NoSuchAlgorithmException, InvalidKeyException, BadPaddingException,
IllegalBlockSizeException {
        Cipher cipher = Cipher.getInstance("RSA/ECB/PKCS1Padding");
        cipher.init(Cipher.DECRYPT_MODE, privateKey);
        return new String(cipher.doFinal(data));
    }

    public static String decrypt(String data, String base64PrivateKey) throws
IllegalBlockSizeException,
            InvalidKeyException, BadPaddingException, NoSuchAlgorithmException,
NoSuchPaddingException {
        return decrypt(Base64.getDecoder().decode(data.getBytes()),
getPrivateKey(base64PrivateKey));
    }

    public static void main(String[] args) throws IllegalBlockSizeException, InvalidKeyException,
            NoSuchPaddingException, BadPaddingException, IOException {
```

```
        try {
            ServerSocket ss = new ServerSocket(6666);
            Socket s = ss.accept();// establishes connection
            DataInputStream dis = new DataInputStream(s.getInputStream());
            String encryptedString = (String) dis.readUTF();
            System.out.println("Encrypted String is:\n" + encryptedString);

            String decryptedString = Server.decrypt(encryptedString, privateKey);
            System.out.println("Decrypted String is:\n" + decryptedString);

        } catch (Exception e) {
        }
        ;
    }
}
```

Output:

```
shankar@shankar-ThinkPad-L450:~/Documents/AU/sem6/security/lab/week7/AES$ java Client.java
Enter the string:hello there

Encryption String is:
WZIZdDSA2pigJqX90WFdDnugqqvEIK2+LmRbZdGWc0EaHveGGgKpcpeJGrNaqHmqTBwvURNMpMG+ZoJY4h2iJc/3b/VHCL9EB5YLmSk/Y7DpTPvlxP+XOPErLZwXTrx8VKkD8gnoBPgCB3+R40NvDe
M8w8UEiBcY+3y1xbyTg3w=
```

```
shankar@shankar-ThinkPad-L450:~/Documents/AU/sem6/security/lab/week7/AES$ java Server.java
Encrypted String is:
WZIZdDSA2pigJqX90WFdDnugqqvEIK2+LmRbZdGWc0EaHveGGgKpcpeJGrNaqHmqTBwvURNMpMG+ZoJY4h2iJc/3b/VHCL9EB5YLmSk/Y7DpTPvlxP+XOPErLZwXTrx8VKkD8gnoBPgCB3+R40NvDe
M8w8UEiBcY+3y1xbyTg3w=
Decrypted String is:
hello there
```