**Assignment-1**

**1.”Caesar Cipher”**

import java.util.Scanner;

class CaesarCypher{

public static void main(String args[]){

Scanner in = new Scanner(System.in);

String plain\_text;

System.out.print("\nEnter text to be encrypted:");

plain\_text=in.nextLine();

String encrypted\_Text=encryptText(plain\_text);

System.out.print("\nEncrypted Text:"+encrypted\_Text);

System.out.print("\nDecrypted Text:"+decryptText(encrypted\_Text));

}

public static String encryptText(String plain\_text)

{

String encrypted\_Text="";

for(int i=0;i<plain\_text.length();i++){

if(((int)plain\_text.charAt(i)) == 32)

encrypted\_Text=encrypted\_Text+" ";

else{

int n=(((int)plain\_text.charAt(i))-65-3);

if(n<0)

n=26-Math.abs(n);

encrypted\_Text=encrypted\_Text+((char)(n+65));

}

}

return encrypted\_Text;

}

public static String decryptText(String plain\_text)

{

String encrypted\_Text="";

for(int i=0;i<plain\_text.length();i++){

if(((int)plain\_text.charAt(i)) == 32)

encrypted\_Text=encrypted\_Text+" ";

else{

int n=(((int)plain\_text.charAt(i))-65+3);

if(n>25)

n=Math.abs(n)-26;

encrypted\_Text=encrypted\_Text+((char)(n+65));

}

}

return encrypted\_Text;

}

}

/\*Output:

Enter text to be encrypted:THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG

Encrypted Text:QEB NRFZH YOLTK CLU GRJMP LSBO QEB IXWV ALD

Decrypted Text:THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG\*/

**2.”DES”**

import java.security.InvalidKeyException;

import java.security.NoSuchAlgorithmException;

import javax.crypto.Cipher;

import javax.crypto.KeyGenerator;

import javax.crypto.NoSuchPaddingException;

import javax.crypto.SecretKey;

import java.util.Base64;

public class DESString {

private static Cipher ecipher;

private static Cipher dcipher;

private static SecretKey key;

public static void main(String[] args) {

try {

// generate secret key using DES algorithm

key = KeyGenerator.getInstance("DES").generateKey();

ecipher = Cipher.getInstance("DES");

dcipher = Cipher.getInstance("DES");

// initialize the ciphers with the given key

ecipher.init(Cipher.ENCRYPT\_MODE, key);

dcipher.init(Cipher.DECRYPT\_MODE, key);

String original="the quick brown fox jumps over a lazy dog!!";

String encrypted = encrypt(original);

System.out.print("Original text:"+original);

System.out.println("\nEncrypted: " + encrypted);

String decrypted = decrypt(encrypted);

System.out.println("\nDecrypted: " + decrypted);

}

catch (Exception e) {

System.out.println("Exception occured!" + e.getMessage());

return;

}

}

public static String encrypt(String str) {

try {

// encode the string into a sequence of bytes using the named charset

byte[] string\_bytes= str.getBytes("UTF8");

// storing the result into a new byte array.

byte[] enc = ecipher.doFinal(string\_bytes);

// encode to base64

enc = Base64.getEncoder().encode(enc);

return new String(enc);

}

catch (Exception e) {

e.printStackTrace();

}

return null;

}

public static String decrypt(String str) {

try {

// decode with base64 to get bytes

byte[] dec = Base64.getDecoder().decode(str.getBytes());

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

return new String(utf8, "UTF8");

}

catch (Exception e) {

e.printStackTrace();

}

return null;

}

}

/\*Output:

D:\Semester-5-\Network security\Assignment 1>javac DESString.java

D:\Semester-5-\Network security\Assignment 1>java DESString

Original text:the quick brown fox jumps over a lazy dog!!

Encrypted: OjizH7l/ytY90hUHfnmBG3GNDavJKuke2AZBmq/6MRY1PozpF6MxpZij9fpEDJFc

Decrypted: the quick brown fox jumps over a lazy dog!!\*/

**3.”MD5”**

import java.math.BigInteger;

import java.security.MessageDigest;

import java.security.NoSuchAlgorithmException;

public class MD5 {

public static void main(String args[]) throws NoSuchAlgorithmException {

String message = "Ankita";

System.out.println("HashCode:" + getMessagedigest(message));

}

public static String getMessagedigest(String message)

{

try {

MessageDigest md = MessageDigest.getInstance("MD5");

byte[] messageDigest = md.digest(message.getBytes());

BigInteger no = new BigInteger(1, messageDigest);

String hashtext = no.toString(16);

while (hashtext.length() < 32) {

hashtext = "0" + hashtext;

}

return hashtext;

}

catch (NoSuchAlgorithmException e) {

throw new RuntimeException(e);

}

}

}

/\*Output

D:\Semester-5-\Network security\Assignment 1>javac MD5.java

D:\Semester-5-\Network security\Assignment 1>java MD5

HashCode:d265e24340d83487e7740d67927e4003\*/

**4.”One Time Pad”**

import java.util.Scanner;

import java.util.Random;

class OneTimePad{

public static void main(String args[]){

Scanner in = new Scanner(System.in);

String plain\_text;

StringBuffer key=new StringBuffer();

StringBuffer encrypted\_text=new StringBuffer();

StringBuffer decrypted\_text=new StringBuffer();

System.out.print("\nEnter plain text:");

plain\_text=in.nextLine();

key=generateKey(plain\_text);

System.out.print("\nKey:"+key);

encrypted\_text=encryptText(plain\_text,key);

System.out.print("\nEncrypted text is :"+encrypted\_text);

decrypted\_text=decryptText(encrypted\_text,key);

System.out.print("\nDecrypted text is :"+decrypted\_text);

}

public static StringBuffer generateKey(String plain\_text)

{

Random random = new Random();

StringBuffer key=new StringBuffer("");

for(int i=0;i<plain\_text.length();i++)

{

char a=(char)(random.nextInt(128));

key.append((char)(a));

}

return key;

}

public static StringBuffer encryptText(String plain\_text,StringBuffer key)

{

StringBuffer encrypted\_text=new StringBuffer("");

for(int i=0;i<plain\_text.length();i++)

{

encrypted\_text.append((char)(((int)(plain\_text.charAt(i)))^((int)(key.charAt(i)))));

}

return encrypted\_text;

}

public static StringBuffer decryptText(StringBuffer encrypted\_text,StringBuffer key)

{

StringBuffer decrypted\_text=new StringBuffer("");

for(int i=0;i<encrypted\_text.length();i++)

{

decrypted\_text.append((char)(((int)(encrypted\_text.charAt(i)))^((int)(key.charAt(i)))));

}

return decrypted\_text;

}

}

/\*Output:

C:\Users\user\Desktop\Sem 5\Semester-5-\Network security>javac OneTimePad.java

C:\Users\user\Desktop\Sem 5\Semester-5-\Network security>java OneTimePad

Enter plain text:Antra koul

Key:><"~

**5.”RSA “**

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 = "My name is Antra!";

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/OAEPWITHSHA-512ANDMGF1PADDING");

//Initialize Cipher for DECRYPT\_MODE

cipher.init(Cipher.DECRYPT\_MODE, privateKey);

//Perform Decryption

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

return new String(decryptedTextArray);

}

}

/\*Output:

D:\Semester-5-\Network security\Assignment 1>javac RSAEncryption.java

D:\Semester-5-\Network security\Assignment 1>java RSAEncryption

Original Text : My name is Antra!

Encrypted Text : 

DeCrypted Text : My name is Antra!\*/

**6.”SHA1”**

import java.math.BigInteger;

import java.security.MessageDigest;

import java.security.NoSuchAlgorithmException;

public class SHA1 {

public static void main(String args[]) throws NoSuchAlgorithmException {

String message = "Ankita";

System.out.println("HashCode:" + getMessagedigest(message));

}

public static String getMessagedigest(String message)

{

try {

MessageDigest md = MessageDigest.getInstance("SHA1");

byte[] messageDigest = md.digest(message.getBytes());

BigInteger no = new BigInteger(1, messageDigest);

String hashtext = no.toString(16);

while (hashtext.length() < 32) {

hashtext = "0" + hashtext;

}

return hashtext;

}

catch (NoSuchAlgorithmException e) {

throw new RuntimeException(e);

}

}

}

/\*Output:

D:\Semester-5-\Network security\Assignment 1>javac SHA1.java

D:\Semester-5-\Network security\Assignment 1>java SHA1

HashCode:7aa4ca80d0edd35cb05f88b5a7bd4cf228af3f21\*/

#Xx

C^eL-ted text is :NPH

Decrypted text is :Antra koul

\*/

**7.”Substitution cipher”**

import java.util.Scanner;

import java.io.\*;

class Substitution{

public static void main(String args[]){

Scanner in = new Scanner(System.in);

String plain\_text;

try{

BufferedReader br = new BufferedReader(new FileReader(new File("SubstitutionKey.txt")));

int key=Integer.parseInt(br.readLine());

System.out.print("\nEnter text to be encrypted:");

plain\_text=in.nextLine();

String encrypted\_Text=encryptText(plain\_text,key);

System.out.print("\nEncrypted Text:"+encrypted\_Text);

System.out.print("\nDecrypted Text:"+decryptText(encrypted\_Text,key));

}

catch(Exception e){

e.printStackTrace();

}

}

public static String encryptText(String plain\_text,int key)

{

String encrypted\_Text="";

for(int i=0;i<plain\_text.length();i++){

if(((int)plain\_text.charAt(i)) == 32)

encrypted\_Text=encrypted\_Text+" ";

else{

int n=(((int)plain\_text.charAt(i))-65-key);

if(n<0)

n=26-Math.abs(n);

encrypted\_Text=encrypted\_Text+((char)(n+65));

}

}

return encrypted\_Text;

}

public static String decryptText(String plain\_text,int key)

{

String encrypted\_Text="";

for(int i=0;i<plain\_text.length();i++){

if(((int)plain\_text.charAt(i)) == 32)

encrypted\_Text=encrypted\_Text+" ";

else{

int n=(((int)plain\_text.charAt(i))-65+key);

if(n>25)

n=Math.abs(n)-26;

encrypted\_Text=encrypted\_Text+((char)(n+65));

}

}

return encrypted\_Text;

}

}

/\*Output:

C:\Users\user\Desktop\Sem 5\Network security>javac Substitution.java

C:\Users\user\Desktop\Sem 5\Network security>java Substitution

Enter text to be encrypted:THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG

Encrypted Text:QEB NRFZH YOLTK CLU GRJMP LSBO QEB IXWV ALD

Decrypted Text:THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG

\*/

Substitution key

3

**8.”Sunstitution Client Server:”**

import java.net.\*;

import java.io.\*;

import java.util.\*;

class SubstitutionUDP

{

public static void main(String args[])throws Exception

{

DatagramSocket datagram\_socket = new DatagramSocket();

DatagramPacket datagram\_packet = null;

InetAddress ip = InetAddress.getLocalHost();

Scanner in = new Scanner(System.in);

byte[] send\_packet = new byte[65535];

byte[] receive\_packet = new byte[65535];

BufferedReader br = new BufferedReader(new FileReader(new File("SubstitutionKey.txt")));

int key=Integer.parseInt(br.readLine());

while(true)

{

String message\_string ;

System.out.print("\nClient:");

message\_string= in.nextLine();

send\_packet = message\_string.getBytes();

datagram\_packet = new DatagramPacket(send\_packet,send\_packet.length,ip,1234);

datagram\_socket.send(datagram\_packet);

datagram\_packet = new DatagramPacket(receive\_packet,receive\_packet.length);

datagram\_socket.receive(datagram\_packet);

message\_string = convert\_to\_string(receive\_packet);

System.out.print("Server : "+message\_string);

System.out.print("\nDecrypted Text:"+decryptText(message\_string,key));

if(message\_string.equals("EXIT") || message\_string.equals("BYE"))

{

System.out.print("Client exiting.............");

break;

}

send\_packet = new byte[65535];

receive\_packet = new byte[65535];

}

}

public static String convert\_to\_string(byte[] buffer)

{

if ( buffer == null )

{

System.out.print("\n No message sent !");

return null;

}

String message = "";

int i = 0;

while ( buffer[i] != 0 )

{

message = message + (char)buffer[i];

i++;

}

return message;

}

public static String decryptText(String plain\_text,int key)

{

String encrypted\_Text="";

for(int i=0;i<plain\_text.length();i++){

if(((int)plain\_text.charAt(i)) == 32)

encrypted\_Text=encrypted\_Text+" ";

else{

int n=(((int)plain\_text.charAt(i))-65+key);

if(n>25)

n=Math.abs(n)-26;

encrypted\_Text=encrypted\_Text+((char)(n+65));

}

}

return encrypted\_Text;

}

}

import java.io.\*;

import java.net.\*;

import java.util.\*;

class SubstitutionServer

{

public static void main(String args[])throws Exception

{

DatagramSocket datagram\_socket = new DatagramSocket(1234);

InetAddress ip = InetAddress.getLocalHost();

byte[] recieve\_packet = new byte[65535];

byte[] send\_packet = new byte[65535];

DatagramPacket datagram\_packet = null;

BufferedReader br = new BufferedReader(new FileReader(new File("SubstitutionKey.txt")));

int key=Integer.parseInt(br.readLine());

while(true)

{

datagram\_packet = new DatagramPacket(recieve\_packet , recieve\_packet.length);

datagram\_socket.receive(datagram\_packet);

String message = convert\_to\_String(recieve\_packet);

System.out.print("\nClient's Plain text:"+message);

message=encryptText(message,key);

System.out.print("\nServer Encrypted text:"+message);

send\_packet = message.getBytes();

ip = datagram\_packet.getAddress();

int port = datagram\_packet.getPort();

datagram\_packet = new DatagramPacket(send\_packet, send\_packet.length , ip , port);

datagram\_socket.send(datagram\_packet);

if(message.equals("EXIT") || message.equals("BYE"))

{

System.out.print("\n Client exiting!");

datagram\_socket.close();

break;

}

recieve\_packet = new byte[65535];

send\_packet = new byte[65535];

}

}

public static String convert\_to\_String(byte[] buf)

{

if(buf == null)

{

System.out.print("\n No message recieved !");

return null;

}

String message = "";

int i = 0;

while ( buf[i] != 0 )

{

message = message + (char)buf[i];

i++;

}

return message;

}

public static String encryptText(String plain\_text,int key)

{

String encrypted\_Text="";

for(int i=0;i<plain\_text.length();i++){

if(((int)plain\_text.charAt(i)) == 32)

encrypted\_Text=encrypted\_Text+" ";

else{

int n=(((int)plain\_text.charAt(i))-65-key);

if(n<0)

n=26-Math.abs(n);

encrypted\_Text=encrypted\_Text+((char)(n+65));

}

}

return encrypted\_Text;

}

}

/\*Output:

C:\Users\user\Desktop\Sem 5\Network security>javac SubstitutionServer.java

C:\Users\user\Desktop\Sem 5\Network security>java SubstitutionServer

Client's Plain text:THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG

Server Encrypted text:QEB NRFZH YOLTK CLU GRJMP LSBO QEB IXWV ALD

C:\Users\user\Desktop\Sem 5\Network security>javac SubstitutionUDP.java

C:\Users\user\Desktop\Sem 5\Network security>java SubstitutionUDP

Client:THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG

Server : QEB NRFZH YOLTK CLU GRJMP LSBO QEB IXWV ALD

Decrypted Text:THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG

Client:

**\*/**

**9.”Transposition Cipher”**

import java.util.Scanner;

import java.util.TreeMap;

import java.util.Iterator;

import java.util.Map;

import java.util.ArrayList;

import java.io.File;

import java.io.FileReader;

import java.io.BufferedReader;

class TranspositionCypher{

public static void main(String args[]){

Scanner in = new Scanner(System.in);

System.out.print("Enter the text to be encrypted:");

String plain\_text=in.nextLine();

try{

BufferedReader br = new BufferedReader(new FileReader(new File("TranspositionKey.txt")));

String key = br.readLine();

TreeMap<String,String> encrypt\_text = new TreeMap<String,String>();

encrypt\_text=insert\_key(key,encrypt\_text,plain\_text);

System.out.print("\nEncrypted text:");

int len=printEncryptedText(encrypt\_text);

System.out.print("\nDecrypted text:");

printDecryptedText(encrypt\_text,key,len);

}

catch(Exception e){

System.out.print("Exception occured!"+e.getMessage());

}

}

public static void printDecryptedText(TreeMap<String,String> encrypt\_text,String key,int length)

{

String str[] = key.split("");

int counter=0,i=0;

ArrayList<String> text = new ArrayList<String>();

for(String s:str){

text.add(encrypt\_text.get(s));

}

while(counter<length)

{

for(int j=0;j<text.size();j++)

{

String s=text.get(j);

if(i<s.length()){

System.out.print(s.charAt(i));

counter++;

}

}

i++;

}

}

public static int printEncryptedText(TreeMap<String,String> encrypt\_text)

{

int count=0;

Iterator itr =encrypt\_text.entrySet().iterator();

while (itr.hasNext()) {

Map.Entry mapElement = (Map.Entry)itr.next();

String str=mapElement.getValue().toString();

System.out.print(str);

count=count+str.length();

}

return count;

}

public static TreeMap<String,String> insert\_key(String key, TreeMap<String,String>encrypt\_text,String plain\_text)

{

String[] str = key.split("");

String text="";

int i,j,key\_length,text\_length;

j=0;

key\_length=key.length();

text\_length=plain\_text.length();

for(String s:str){

i=j;

while(i<text\_length){

text=text+(plain\_text.charAt(i));

if(i+key\_length < text\_length){

i+=key\_length;

}

else

i=text\_length;

}

encrypt\_text.put(s,text);

j+=1;

text="";

}

return encrypt\_text;

}

}

/\*Output:

C:\Users\user\Desktop\Sem 5\Network security>javac TranspositionCypher.java

C:\Users\user\Desktop\Sem 5\Network security>java TranspositionCypher

Enter the text to be encrypted:please transfer one million dollars to my swiss bank account six two two

Encrypted text:as wktosfmdti rll sciwlanor autenenssnnwt llm cxoproiaybo eeioosast

Decrypted text:please transfer one million dollars to my swiss bank account six two two\*/

Transposition key:

MEGABUCK

10**.”AES”**

import javax.crypto.\*;

import javax.crypto.spec.IvParameterSpec;

import javax.crypto.spec.SecretKeySpec;

class AES{

static String Iv="AAAAAAAAAAAAAAAA";

static String plain\_text="test text 123\0\0\0";

static String key= "0123456789abcdef";

public static void main(String args[]){

try{

System.out.print("\nPlain text:"+plain\_text);

byte[] encrypted\_text=encrypt(plain\_text,key);

System.out.print("\nEncrypted text:");

for(int i=0;i<encrypted\_text.length;i++){

System.out.print(new Integer(encrypted\_text[i]+""));

}

String decrypted\_text=decrypt(encrypted\_text,key);

System.out.print("\nDecrypted text:"+decrypted\_text);

}

catch(Throwable e){

e.printStackTrace();

}

}

public static byte[] encrypt(String plain\_text,String key)throws Throwable{

Cipher cipher=Cipher.getInstance("AES/CBC/NoPadding", "SunJCE");

SecretKeySpec secret\_key\_spec = new SecretKeySpec(key.getBytes("UTF-8"),"AES");

cipher.init(Cipher.ENCRYPT\_MODE,secret\_key\_spec,new IvParameterSpec(Iv.getBytes("UTF-8")));

return cipher.doFinal(plain\_text.getBytes());

}

public static String decrypt(byte[] cipher\_text,String key)throws Throwable{

Cipher cipher=Cipher.getInstance("AES/CBC/NoPadding", "SunJCE");

SecretKeySpec secret\_key\_spec = new SecretKeySpec(key.getBytes("UTF-8"),"AES");

cipher.init(Cipher.DECRYPT\_MODE,secret\_key\_spec,new IvParameterSpec(Iv.getBytes("UTF-8")));

return new String(cipher.doFinal(cipher\_text),"UTF-8");

}

}

/\*Output:

C:\Users\user\Desktop\Sem 5\Semester-5-\Network security\AES>javac AES.java

Note: AES.java uses or overrides a deprecated API.

Note: Recompile with -Xlint:deprecation for details.

C:\Users\user\Desktop\Sem 5\Semester-5-\Network security\AES>java AES

Plain text:test text 123

Encrypted text:16-12441-83-16-12361-64-15-74872863306478

Decrypted text:test text 123\*/

**“Assignment2”**

**1.** **Implement authentication Service. The sender sends password in encrypted format to the receiver, the receiver checks**

**the password (after decrypting) in its database/array and replies back as success or failure.(Keys are already shared)**

**“Client”**

import java.net.DatagramSocket;

import java.net.DatagramPacket;

import java.net.InetAddress;

import java.util.Scanner;

import java.io.IOException;

class Question1Sender

{

public static void main(String args[]) throws IOException

{

DatagramSocket client = new DatagramSocket();

InetAddress ip = InetAddress.getLocalHost();

byte[] send = new byte[65536];

byte[] recieve = new byte[65536];

Scanner in = new Scanner(System.in);

System.out.print("\nEnter username:");

String username=in.nextLine();

System.out.print("\nEnter password:");

String password = in.nextLine();

password = encryptPassword(password);

String data=username+","+password;

send=data.getBytes();

DatagramPacket packet = new DatagramPacket(send,send.length,ip,1234);

client.send(packet);

packet = new DatagramPacket(recieve,recieve.length);

client.receive(packet);

System.out.print("Server:"+convertToString(recieve));

}

public static String convertToString(byte[] a)

{

if (a == null)

return null;

String s = "";

int i = 0;

while (a[i] != 0)

{

s=s+(char)a[i];

i++;

}

return s;

}

public static String encryptPassword(String password){

char[] tokens = password.toCharArray();

String encrypted\_text="";

for(char c:tokens){

encrypted\_text=encrypted\_text+((char)((int)c+3))+"";

}

return encrypted\_text;

}

}

“Server”

import java.net.DatagramSocket;

import java.net.DatagramPacket;

import java.net.InetAddress;

import java.util.Scanner;

import java.io.IOException;

import java.util.HashMap;

class Question1Reciever

{

static HashMap<String,String> user\_data = new HashMap<String,String>();

public static void main(String args[]) throws IOException

{

user\_data.put("ANTRA","ANTRA");

user\_data.put("ANKITA","ANKITA");

DatagramSocket server = new DatagramSocket(1234);

byte[] send = new byte[65536];

byte[] recieve = new byte[65536];

DatagramPacket packet = new DatagramPacket(recieve,recieve.length);

server.receive(packet);

String[] message=(convertToString(recieve)).split(",");

String username=message[0];

String password=decryptPassword(message[1]);

InetAddress ip = packet.getAddress();

int port=packet.getPort();

System.out.print("Username"+username+"Password"+password);

String status=check(password,username);

packet=new DatagramPacket(status.getBytes(),status.getBytes().length,ip,port);

server.send(packet);

}

//authoriziing function

public static String check(String password,String username){

if(password.equals((String)user\_data.get(username))){

return "success";

}

return "failure";

}

//converting bytes to string

public static String convertToString(byte[] a)

{

if (a == null)

return null;

String s = "";

int i = 0;

while (a[i] != 0)

{

s=s+(char)a[i];

i++;

}

return s;

}

//decrypting password

public static String decryptPassword(String password){

char[] tokens = password.toCharArray();

String decrypted\_text="";

for(char c:tokens){

decrypted\_text=decrypted\_text+((char)((int)c-3))+"";

}

return decrypted\_text;

}

}

/\*Output:

PS D:\Semester-5-\Network security\Assignment 2> javac Question1Sender.java

PS D:\Semester-5-\Network security\Assignment 2> java Question1Sender

Enter username:ANKITA

Enter password:ANKITA

Server:success

PS D:\Semester-5-\Network security\Assignment 2> javac Question1Sender.java

PS D:\Semester-5-\Network security\Assignment 2> java Question1Sender

Enter username:ABC

Enter password:ABC

Server:failure

PS D:\Semester-5-\Network security\Assignment 2> javac Question2Reciever.java

PS D:\Semester-5-\Network security\Assignment 2> java Question2Reciever

UsernameANKITAPasswordANKITA

PS D:\Semester-5-\Network security\Assignment 2> javac Question2Reciever.java

PS D:\Semester-5-\Network security\Assignment 2> java Question2Reciever

UsernameABCPasswordABC\*/

2. **Implement authentication Service. The sender sends password in encrypted format to the receiver,**

**the receiver checks the password (after decrypting and applying hash) in its database/array and replies**

**back as success or failure. (Note: Here the password is stored as hash in database).\*/**

**“Client”**

import java.net.DatagramSocket;

import java.net.DatagramPacket;

import java.net.InetAddress;

import java.util.Scanner;

import java.io.IOException;

public class Question2Sender

{

public static void main(String args[]) throws IOException

{

DatagramSocket client = new DatagramSocket();

InetAddress ip = InetAddress.getLocalHost();

byte[] message = new byte[65536];

Scanner in = new Scanner(System.in);

System.out.print("\nEnter username:");

String username=in.nextLine();

System.out.print("\nEnter password:");

String password = in.nextLine();

password = encryptPassword(password);

String data=username+","+password;

message=data.getBytes();

DatagramPacket packet = new DatagramPacket(message,message.length,ip,1234);

client.send(packet);

message = new byte[65536];

packet = new DatagramPacket(message,message.length);

client.receive(packet);

System.out.print("Server:"+convertToString(message));

}

public static String convertToString(byte[] a)

{

if (a == null)

return null;

String s = "";

int i = 0;

while (a[i] != 0)

{

s=s+(char)a[i];

i++;

}

return s;

}

public static String encryptPassword(String password){

char[] tokens = password.toCharArray();

String encrypted\_text="";

for(char c:tokens){

encrypted\_text=encrypted\_text+((char)((int)c+3))+"";

}

return encrypted\_text;

}

}

**“Server”**

import java.net.DatagramSocket;

import java.net.DatagramPacket;

import java.net.InetAddress;

import java.util.Scanner;

import java.io.IOException;

import java.util.HashMap;

import java.math.BigInteger;

import java.security.MessageDigest;

import java.security.NoSuchAlgorithmException;

class Question2Receiver

{

static HashMap<String,String> user\_data = new HashMap<String,String>();

public static void main(String args[]) throws IOException

{

user\_data.put("Antra","182c4b89a27c52474a0a532d5519f5e1");

user\_data.put("Ankita","d265e24340d83487e7740d67927e4003");

DatagramSocket server = new DatagramSocket(1234);

byte[] message = new byte[65536];

DatagramPacket packet = new DatagramPacket(message,message.length);

server.receive(packet);

InetAddress ip=packet.getAddress();

int port=packet.getPort();

String[] str=(convertToString(message)).split(",");

String username=str[0];

String password=str[1];

password=decryptPassword(password);

System.out.print("Username:"+username+"Password:"+password);

message=null;

message=(check(password,username)).getBytes();

packet=new DatagramPacket(message,message.length,ip,port);

server.send(packet);

}

//authoriziing function

public static String check(String password,String username){

if((getMessagedigest(password)).equals((String)user\_data.get(username))){

return "success";

}

return "failure";

}

//converting bytes to string

public static String convertToString(byte[] a)

{

if (a == null)

return null;

String s = "";

int i = 0;

while (a[i] != 0)

{

s=s+(char)a[i];

i++;

}

return s;

}

//decrypting password

public static String decryptPassword(String password){

char[] tokens = password.toCharArray();

String decrypted\_text="";

for(char c:tokens){

decrypted\_text=decrypted\_text+((char)((int)c-3))+"";

}

return decrypted\_text;

}

public static String getMessagedigest(String message)

{

try {

MessageDigest md = MessageDigest.getInstance("MD5");

byte[] messageDigest = md.digest(message.getBytes());

BigInteger no = new BigInteger(1, messageDigest);

String hashtext = no.toString(16);

while (hashtext.length() < 32) {

hashtext = "0" + hashtext;

}

return hashtext;

}

catch (NoSuchAlgorithmException e) {

throw new RuntimeException(e);

}

}

}

/\*Output:

PS D:\Semester-5-\Network security\Assignment 2> javac Question2Sender.java

PS D:\Semester-5-\Network security\Assignment 2> java Question2Sender

Enter username:Antra

Enter password:ANtra

Server:failure

PS D:\Semester-5-\Network security\Assignment 2> javac Question2Sender.java

PS D:\Semester-5-\Network security\Assignment 2> java Question2Sender

Enter username:Antra

Enter password:Antra

Server:success

PS D:\Semester-5-\Network security\Assignment 2> javac Question2Receiver.java

PS D:\Semester-5-\Network security\Assignment 2> java Question2Receiver

Username:AntraPassword:ANtra

PS D:\Semester-5-\Network security\Assignment 2> javac Question2Receiver.java

PS D:\Semester-5-\Network security\Assignment 2> java Question2Receiver

Username:AntraPassword:Antra\*/

**3.** **Implement a firewall that behaves like forwarder. It does not forward the packet that contains the word "terrorist".**

**“Client”**

import java.net.InetAddress;

import java.util.Scanner;

import java.io.IOException;

import java.net.DatagramSocket;

import java.net.DatagramPacket;

class Question3Client

{

public static void main(String args[]) throws IOException

{

DatagramSocket client = new DatagramSocket();

DatagramPacket packet = null;

InetAddress ip = InetAddress.getLocalHost();

byte[] packetBytes = new byte[65536];

Scanner in = new Scanner(System.in);

String message="";

while(!message.equalsIgnoreCase("bye")){

System.out.print("\nClient:");

message = in.nextLine();

if(message.equalsIgnoreCase("bye"))

{

System.out.print("\nConversation ended!");

break;

}

packetBytes = message.getBytes();

packet = new DatagramPacket(packetBytes,packetBytes.length,ip,1234);

client.send(packet);

packetBytes = new byte[65536];

packet = new DatagramPacket(packetBytes,packetBytes.length);

client.receive(packet);

System.out.print("Server:"+convertToString(packetBytes));

}

}

public static String convertToString(byte[] a)

{

if (a == null)

return null;

String s = "";

int i = 0;

while (a[i] != 0)

{

s=s+(char)a[i];

i++;

}

return s;

}

}

**“Firewall”**

import java.net.DatagramSocket;

import java.net.DatagramPacket;

import java.net.InetAddress;

import java.util.Scanner;

import java.io.IOException;

import java.util.HashMap;

class Question3Firewall

{

public static void main(String args[]) throws IOException

{

DatagramSocket server = new DatagramSocket(1234);

byte[] packetBytes = new byte[65536];

String message="",status;

InetAddress ip;

int port;

DatagramPacket packet=null;

while(!message.equalsIgnoreCase("bye")){

packet=new DatagramPacket(packetBytes,packetBytes.length);

server.receive(packet);

System.out.print("\nClient:"+message);

message=convertToString(packetBytes);

if(message.equalsIgnoreCase("bye")){

System.out.print("\n Connection terminated!");

break;

}

status=check(message);

if(status.equals("not sent!")){

ip = packet.getAddress();

port=packet.getPort();

packetBytes=status.getBytes();

packet=new DatagramPacket(packetBytes,packetBytes.length,ip,port);

server.send(packet);

}

else

{

ip = packet.getAddress();

port=packet.getPort();

packetBytes=status.getBytes();

packet=new DatagramPacket(packetBytes,packetBytes.length,ip,port);

server.send(packet);

packetBytes = new byte[65536];

ip=InetAddress.getLocalHost();

packetBytes=message.getBytes();

packet=new DatagramPacket(packetBytes,packetBytes.length,ip,1111);

server.send(packet);

}

packetBytes = new byte[65536];

}

}

//authoriziing function

public static String check(String message){

String[] tokens = message.split(" ");

for(String s:tokens){

if(s.equalsIgnoreCase("terrorist"))

{

System.out.println("Can't be forwarded!");

return "not sent!";

}

}

return "sent!";

}

//converting bytes to string

public static String convertToString(byte[] a)

{

if (a == null)

return null;

String s = "";

int i = 0;

while (a[i] != 0)

{

s=s+(char)a[i];

i++;

}

return s;

}

}

**“Server”**

import java.net.DatagramSocket;

import java.net.DatagramPacket;

import java.net.InetAddress;

import java.util.Scanner;

import java.io.IOException;

import java.util.HashMap;

class Question3Server

{

public static void main(String args[]) throws IOException

{

DatagramSocket server = new DatagramSocket(1111);

byte[] packetBytes = new byte[65536];

String message="";

DatagramPacket packet=null;

while(!message.equalsIgnoreCase("bye")){

packet=new DatagramPacket(packetBytes,packetBytes.length);

server.receive(packet);

message=convertToString(packetBytes);

System.out.print("\nClient:"+message);

if(message.equalsIgnoreCase("bye")){

System.out.print("\n Connection terminated!");

break;

}

packetBytes = new byte[65536];

}

}

//converting bytes to string

public static String convertToString(byte[] a)

{

if (a == null)

return null;

String s = "";

int i = 0;

while (a[i] != 0)

{

s=s+(char)a[i];

i++;

}

return s;

}

}

/\*Output:

PS D:\Semester-5-\Network security\Assignment 2> javac Question3Client.java

PS D:\Semester-5-\Network security\Assignment 2> java Question3Client

Client:Hello

Server:sent!

Client:How are you?

Server:sent!

Client:terrorist

Server:not sent!

Client:Hey again?

Server:sent!

Client:

PS D:\Semester-5-\Network security\Assignment 2> javac Question3Firewall.java

PS D:\Semester-5-\Network security\Assignment 2> java Question3Firewall

Client:

Client:Hello

Client:How are you?Can't be forwarded!

Client:terrorist

PS D:\Semester-5-\Network security\Assignment 2> javac Question3Server.java

PS D:\Semester-5-\Network security\Assignment 2> java Question3Server

Client:Hello

Client:How are you?

Client:Hey again?\*/

4. **Implement NAT functionality. The NAT works like forwarder, that will forward to appropriate receiver.**

**“Sender”**

import java.net.InetAddress;

import java.util.Scanner;

import java.io.IOException;

import java.net.DatagramSocket;

import java.net.DatagramPacket;

class Question4Sender

{

public static void main(String args[]) throws IOException

{

DatagramSocket client = new DatagramSocket();

DatagramPacket packet = null;

InetAddress ip = InetAddress.getLocalHost();

byte[] packetBytes = new byte[65536];

Scanner in = new Scanner(System.in);

String message="";

while(!message.equalsIgnoreCase("bye")){

System.out.print("\nClient:");

message = in.nextLine();

if(message.equalsIgnoreCase("bye"))

{

System.out.print("\nConversation ended!");

break;

}

packetBytes = message.getBytes();

packet = new DatagramPacket(packetBytes,packetBytes.length,ip,1234);

client.send(packet);

}

}

public static String convertToString(byte[] a)

{

if (a == null)

return null;

String s = "";

int i = 0;

while (a[i] != 0)

{

s=s+(char)a[i];

i++;

}

return s;

}

}

**“NAT”**

import java.net.DatagramSocket;

import java.net.DatagramPacket;

import java.net.InetAddress;

import java.io.IOException;

class Question4NAT

{

public static void main(String args[]) throws IOException

{

DatagramSocket server = new DatagramSocket(1234);

byte[] packetBytes = new byte[65536];

String message="",status;

InetAddress ip;

int port;

DatagramPacket packet=null;

while(!message.equalsIgnoreCase("bye")){

packet=new DatagramPacket(packetBytes,packetBytes.length);

server.receive(packet);

message=convertToString(packetBytes);

System.out.print("\nClient:"+message);

if(message.equalsIgnoreCase("bye")){

System.out.print("\n Connection terminated!");

break;

}

status=check(message);

ip=InetAddress.getLocalHost();

packetBytes=message.getBytes();

if(status.equals("even")){

packet=new DatagramPacket(packetBytes,packetBytes.length,ip,1111);

server.send(packet);

}

else if(status.equals("odd"))

{

packet=new DatagramPacket(packetBytes,packetBytes.length,ip,2222);

server.send(packet);

}

else{

System.out.print("\n Invalid message!");

}

packetBytes = new byte[65536];

}

}

//authoriziing function

public static String check(String message){

if((Integer.parseInt(message)) % 2 == 0)

return "even";

else if((Integer.parseInt(message)) % 2 != 0)

return "odd";

else

return "failure!";

}

//converting bytes to string

public static String convertToString(byte[] a)

{

if (a == null)

return null;

String s = "";

int i = 0;

while (a[i] != 0)

{

s=s+(char)a[i];

i++;

}

return s;

}

}

**“Receiver1”**

import java.net.DatagramSocket;

import java.net.DatagramPacket;

import java.net.InetAddress;

import java.io.IOException;

class Question4Server1

{

public static void main(String args[]) throws IOException

{

DatagramSocket server = new DatagramSocket(1111);

byte[] packetBytes = new byte[65536];

String message="";

DatagramPacket packet=null;

while(!message.equalsIgnoreCase("bye")){

packet=new DatagramPacket(packetBytes,packetBytes.length);

server.receive(packet);

message=convertToString(packetBytes);

System.out.print("\nClient:"+message);

if(message.equalsIgnoreCase("bye")){

System.out.print("\n Connection terminated!");

break;

}

packetBytes = new byte[65536];

}

}

//converting bytes to string

public static String convertToString(byte[] a)

{

if (a == null)

return null;

String s = "";

int i = 0;

while (a[i] != 0)

{

s=s+(char)a[i];

i++;

}

return s;

}

}

**“Receiver2”**

import java.net.DatagramSocket;

import java.net.DatagramPacket;

import java.net.InetAddress;

import java.io.IOException;

class Question4Server2

{

public static void main(String args[]) throws IOException

{

DatagramSocket server = new DatagramSocket(2222);

byte[] packetBytes = new byte[65536];

String message="";

DatagramPacket packet=null;

while(!message.equalsIgnoreCase("bye")){

packet=new DatagramPacket(packetBytes,packetBytes.length);

server.receive(packet);

message=convertToString(packetBytes);

System.out.print("\nClient:"+message);

if(message.equalsIgnoreCase("bye")){

System.out.print("\n Connection terminated!");

break;

}

packetBytes = new byte[65536];

}

}

//converting bytes to string

public static String convertToString(byte[] a)

{

if (a == null)

return null;

String s = "";

int i = 0;

while (a[i] != 0)

{

s=s+(char)a[i];

i++;

}

return s;

}

}

/\*Output:

PS D:\Semester-5-\Network security\Assignment 2> javac Question4Sender.java

PS D:\Semester-5-\Network security\Assignment 2> java Question4Sender

Client:2

Client:23

Client:32

Client:11

Client:

PS D:\Semester-5-\Network security\Assignment 2> javac Question4NAT.java

PS D:\Semester-5-\Network security\Assignment 2> java Question4NAT

Client:2

Client:23

Client:32

Client:11

PS D:\Semester-5-\Network security\Assignment 2> javac Question4Server1.java

PS D:\Semester-5-\Network security\Assignment 2> java Question4Server1

Client:2

Client:32

PS D:\Semester-5-\Network security\Assignment 2> javac Question4Server2.java

PS D:\Semester-5-\Network security\Assignment 2> java Question4Server2

Client:23

Client:11\*/

5. **Key Distribution**

**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.**

**“KDCSender”**

import java.net.DatagramPacket;

import java.net.DatagramSocket;

import java.net.InetAddress;

import java.util.Scanner;

import java.util.Base64;

import java.security.InvalidKeyException;

import java.security.NoSuchAlgorithmException;

import java.io.\*;

import javax.crypto.Cipher;

import javax.crypto.KeyGenerator;

import javax.crypto.NoSuchPaddingException;

import javax.crypto.SecretKey;

import javax.crypto.spec.SecretKeySpec;

class KDCSender{

public static void main(String args[])throws Exception{

DatagramSocket sender = new DatagramSocket();

DatagramPacket packet = null;

InetAddress ip = InetAddress.getLocalHost();

String requestMessage = "Request for access to server!";

byte[] message= new byte[65536];

message=requestMessage.getBytes();

packet = new DatagramPacket(message,message.length,ip,1234);

sender.send(packet);

message= new byte[65536];

packet = new DatagramPacket(message,message.length);

sender.receive(packet);

String str = convertToString(message);

System.out.print("\nServer:"+str);

BufferedReader br = new BufferedReader(new FileReader(new File("SenderKey.txt")));

String senderKey=br.readLine();

String[] st=(decryptData(str,senderKey)).split(",");

requestMessage=st[0];

message= new byte[65536];

message=requestMessage.getBytes();

packet = new DatagramPacket(message,message.length,ip,2222);

sender.send(packet);

}

public static String decryptData(String message,String senderKey)throws Exception{

SecretKey key;

byte[] keyByte = senderKey.getBytes();

key = new SecretKeySpec(keyByte,0,keyByte.length,"DES");

Cipher ecipher = Cipher.getInstance("DES");

ecipher.init(Cipher.DECRYPT\_MODE, key);

byte[] dec=Base64.getDecoder().decode(message.getBytes());

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

return new String(utf8, "UTF8");

}

public static String convertToString(byte[] a)

{

if (a == null)

return null;

String s = "";

int i = 0;

while (a[i] != 0)

{

s=s+(char)a[i];

i++;

}

return s;

}

}

**“KDC”**

import java.net.DatagramPacket;

import java.net.DatagramSocket;

import java.net.InetAddress;

import java.util.Scanner;

import java.util.Base64;

import java.security.InvalidKeyException;

import java.security.NoSuchAlgorithmException;

import java.io.\*;

import javax.crypto.Cipher;

import javax.crypto.KeyGenerator;

import javax.crypto.NoSuchPaddingException;

import javax.crypto.SecretKey;

import javax.crypto.spec.SecretKeySpec;

class KDC{

public static void main(String args[])throws Exception{

DatagramSocket kdc = new DatagramSocket(1234);

DatagramPacket packet = null;

InetAddress ip ;

byte[] messageBytes=new byte[65536];

String sender="Antra";

String receiver="Ankita";

String sessionKey="Pizza";

String message=sender+","+receiver+","+sessionKey;

BufferedReader br = new BufferedReader(new FileReader(new File("ReceiverKey.txt")));

String receiverKey=br.readLine();

message=encryptData(message,receiverKey);

br = new BufferedReader(new FileReader(new File("SenderKey.txt")));

String senderKey = br.readLine();

message=encryptData(message+","+sessionKey,senderKey);

System.out.print("\nMessage:"+message);

packet = new DatagramPacket(messageBytes,messageBytes.length);

kdc.receive(packet);

String sender\_request=convertToString(messageBytes);

if(sender\_request.equals("Request for access to server!")){

messageBytes=new byte[65536];

messageBytes=message.getBytes();

packet = new DatagramPacket(messageBytes,messageBytes.length,packet.getAddress(),packet.getPort());

kdc.send(packet);

}

}

public static String encryptData(String message,String receiverKey)throws Exception{

SecretKey key;

byte[] keyByte = receiverKey.getBytes();

key = new SecretKeySpec(keyByte,0,keyByte.length,"DES");

Cipher ecipher = Cipher.getInstance("DES");

ecipher.init(Cipher.ENCRYPT\_MODE, key);

byte[] string\_bytes=message.getBytes("UTF-8");

byte[] encoded = ecipher.doFinal(string\_bytes);

encoded = Base64.getEncoder().encode(encoded);

return new String(encoded);

}

public static String convertToString(byte[] a)

{

if (a == null)

return null;

String s = "";

int i = 0;

while (a[i] != 0)

{

s=s+(char)a[i];

i++;

}

return s;

}

}

**“KDCReceiver”**

import java.net.DatagramPacket;

import java.net.DatagramSocket;

import java.net.InetAddress;

import java.util.Scanner;

import java.util.Base64;

import java.security.InvalidKeyException;

import java.security.NoSuchAlgorithmException;

import java.io.\*;

import javax.crypto.Cipher;

import javax.crypto.KeyGenerator;

import javax.crypto.NoSuchPaddingException;

import javax.crypto.SecretKey;

import javax.crypto.spec.SecretKeySpec;

class KDCReceiver{

public static void main(String args[])throws Exception{

DatagramSocket sender = new DatagramSocket(2222);

DatagramPacket packet = null;

byte[] message= new byte[65536];

packet = new DatagramPacket(message,message.length);

sender.receive(packet);

String str = convertToString(message);

BufferedReader br = new BufferedReader(new FileReader(new File("ReceiverKey.txt")));

String senderKey=br.readLine();

str=decryptData(str,senderKey);

System.out.print("\nDecrypted :"+str);

}

public static String decryptData(String message,String receiverKey)throws Exception{

SecretKey key;

byte[] keyByte = receiverKey.getBytes();

key = new SecretKeySpec(keyByte,0,keyByte.length,"DES");

Cipher ecipher = Cipher.getInstance("DES");

ecipher.init(Cipher.DECRYPT\_MODE, key);

byte[] dec=Base64.getDecoder().decode(message.getBytes());

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

return new String(utf8, "UTF8");

}

public static String convertToString(byte[] a)

{

if (a == null)

return null;

String s = "";

int i = 0;

while (a[i] != 0)

{

s=s+(char)a[i];

i++;

}

return s;

}

}

/\*Output:

D:\Semester-5-\Network security\Assignment 2>javac KDCSender.java

D:\Semester-5-\Network security\Assignment 2>java KDCSender

Server:mOdm96LvgHSxIevA/AfCz8CT64IxlWviLf/EgRWP/VECqDpw5fUZfA==

D:\Semester-5-\Network security\Assignment 2>java KDC

Message:mOdm96LvgHSxIevA/AfCz8CT64IxlWviLf/EgRWP/VECqDpw5fUZfA==

D:\Semester-5-\Network security\Assignment 2>javac KDCReceiver.java

D:\Semester-5-\Network security\Assignment 2>java KDCReceiver

Decrypted :Antra,Ankita,Pizza\*/