

SECURITY SYSTEM FOR DNS

By
K.SAMATHA

June 2020

ABSTRACT

The DNS Security is designed by using java programming to provide security by combining the concept of both the Digital Signature and Asymmetric key (Public key) Cryptography. Here the Public key is send instead of Private key.

The DNS security uses Message Digest Algorithm to compress the Message(text file) and PRNG(Pseudo Random Number Generator) Algorithm for generating Public and Private key. The message combines with the Private key to form a Signature using DSA Algorithm, which is send along with the Public key.

The receiver uses the Public key and DSA Algorithm to form a Signature. If this Signature matches with the Signature of the message received, the message is Decrypted and read else discarded.

The Domain Name System(DNS) has become a critical operational part of the Internet Infrastructure, yet it has no strong security mechanisms to assure Data Integrity or Authentication. Extensions to the DNS are described that provide these services to security aware resolves are applications through the use of Cryptographic Digital Signatures. These Digital Signatures are included zones as resource records.

Table of Contents

LIST OF TABLES	v
LIST OF FIGURES	vi
1. INTRODUCTION	1
1.1 GENERAL INTRODUCTION	1
1.2 PROBLEM STATEMENT	1
1.3 SCOPE	2
2. CONTENT	2
2.1 Abbreviations and Acronyms	2
3. LITERATURE SURVEY	2
A. Domain Name System with Security Extensions	2
B. Network Security via Private-Key Certificates	3
C. A new Approach to DNS security (DNSSEC)	3
3.1 EXISTING SYSTEM AND ITS DISADVANTAGE	4
3.2 PROPOSED SYSTEM AND ITS ADVANTAGE	4
4. SYSTEM REQUIREMENTS:	4
5. OVERVIEW OF DNS	5
6. MODULE DESCRIPTION	10
7. IMPLEMENTATION	13
7.1 CODING	13
swing1.java	13
DnsServer12.java	16
SwingRMes.java	31
7.2 SCREENSHOTS:	48
8. Conclusion	58
9. References	58

LIST OF TABLES

CHAPTER 4

Table 1	4
Table 2	4

CHAPTER 5

Table 3	8
---------------	---

LIST OF FIGURES

CHAPTER 5:OVERVIEW OF DNS

Figure 1:Domain name space example	6
Figure 2: Example of inverse domains and the Domain Name Space.....	7
Figure 3: ARCHITECTURE OF DNS:	9
Figure 4: OVERALL DIAGRAM	9
Figure 5:KEY GENERATION	10

CHAPTER 6:MODULE DESCRIPTION

Figure 6:Authentication	10
Figure 7:message encryption	11
Figure 8:signature generation.....	11
Figure 9:verifying signature and decryption	12

CHAPTER 7:IMPLEMENTATION

Figure 10:swing1.java	48
Figure 11: DnsServer12.java	49
Figure 12: SwingRmes.java.....	50
Figure 13: userid and password in screen 1	51
Figure 14: Sender screen.....	51
Figure 15:domain name of the sytem	52
Figure 16 : Sender screen	52
Figure 17: text entry screen	53
Figure 18.....	53
Figure 19: keys creation popup.....	54
Figure 20: Signature generation popup.....	54
Figure 21: system name popup	54
Figure 22: encrypted data.....	55
Figure 23 : Dns server screen	56
Figure 24 : signature verification.....	56
Figure 25: data is sent to the reciever	57

1. INTRODUCTION

1.1 GENERAL INTRODUCTION:

The Domain Name System (DNS) can be considered one of the most important components of the modern Internet. DNS provides a means to map IP addresses (random, hard-to-remember numbers) to names (easier to remember and disseminate).

Without DNS, we would have to remember that `www.amazon.com` is actually the IP address `72.21.207.65`, and that would be hard to change. DNS is really the most successful, largest distributed database. In recent years, however, a number of DNS exploits have been uncovered.

These exploits affect the system in such a way that an end user cannot be certain the mappings he is presented with are in fact legitimate. The DNS Security (DNSSEC) standard has been written in an attempt to mitigate some of the known security issues in the current DNS design used today. Finally, we will analyse the impacts of DNSSEC on embedded platforms and mobile networks.

1.2 PROBLEM STATEMENT

Authenticity is based on the identity of some entity. This entity has to prove that it is genuine. In many Network applications the identity of participating entities is simply determined by their names or addresses. High level applications use mainly names for authentication purposes, because address lists are much harder to create, understand, and maintain than name lists.

Assuming an entity wants to spoof the identity of some other entity, it is enough to change the mapping between its low level address and its high level name. It means that an attacker can fake the name of someone by modifying the association of his address from his own name to the name he

wants to impersonate. Once an attacker has done that, an authenticator can no longer distinguish between the true and fake entity.

1.3 SCOPE:

The Domain Name System(DNS) has become a critical operational part of the Internet Infrastructure, yet it has no strong security mechanisms to assure Data Integrity or Authentication. Extensions to the DNS are described that provide these services to security aware resolves are applications through the use of Cryptographic Digital Signatures. These Digital Signatures are included zones as resource records.

2. CONTENT

2.1 Abbreviations and Acronyms

DNS:

Domain name system

PRNG:

Peudo Random Key Generation algorithm.

DSA:

Digital Signature Algorithm.

3. LITERATURE SURVEY

A. Domain Name System with Security Extensions

This paper presented DNS Wrapper, used to prevent spoofing. The DNS Wrapper examines incoming & outgoing messages. They have implemented one DNS prototype named as „Security Wrapper“. The Security Wrapper is piece of software that encapsulates component such as name server to improve security. The Symmetric Key Cryptography uses the concept of master keys. In this each node shares master key with its parent. The root

has its own Master key & pair of Public and secret keys. To communicate with server it uses Symmetric Certificates. In this method, master key is used to create the certificates which provide safe transfer of keys between levels. The master key is created by root server

B. Network Security via Private-Key Certificates

In this paper, DNS Security solely based on Public key cryptography is used. In public key cryptography, public key, is validated and shared freely with the world, but the private key is kept secret. The key pair is used to encrypt and decrypt data. The data encrypted with the public key can be decrypted only by the corresponding private key.

Public key security system trusts its user to validate each others' public keys & to manage its own private keys security. Public key is widely accepted because it doesn't need trusted key management. This technique is based on validating public key of other. The public cryptography is best suited for securing communication between servers, between sites & organization

C.A new Approach to DNS security (DNSSEC)

In this approach, symmetric key cryptography is used. Minimum messages are send through the network. As it combines two messages i.e. certificate request for authentication and the query messages. Further they have also used hmac (Hash Message Authentication Code) also secret keys, master keys for encryption and decryption of messages.

3.1 EXISTING SYSTEM AND ITS DISADVANTAGE

The existing system does not contain security for sending some data from one system to the other system using the IP address the data or files are sent but the data is not secure

3.2 PROPOSED SYSTEM AND ITS ADVANTAGE

The proposed system has security using Digital signature algorithm the signature is sent from the sender and the receiver verifies the signature and checks whether it matches and then the data is sent from one system to the other.

4. SYSTEM REQUIREMENTS:

SOFTWARE REQUIREMENTS:

OPERATING SYSTEM	WINDOWS XP AND ITS ABOVE VERSIONS
SOFTWARES	JAVA,(NETBEANS)-JDK-1.8.20
DATABASE	NOT NEEDED

Table 1

HARDWARE REQUIREMENTS:

PROCESSOR	INTEL CORE I7
RAM	8GB
SYSTEM TYPE	64-BIT OS

Table 2

Fundamentals of DNS

The DNS not only supports host name to network address resolution, known as forward resolution, but it also supports network address to host name resolution, known as inverse resolution. Due to its ability to map human memorable system names into computer network numerical addresses, its distributed nature, and its robustness, the DNS has evolved into a critical component of the Internet. Without it, the only way to reach other computers on the Internet is to use the numerical network address. Using IP addresses to connect to remote computer systems is not a very user-friendly representation of a system's location on the Internet and thus the DNS is heavily relied upon to retrieve an IP address by just referencing a computer system's Fully Qualified Domain Name (FQDN). A FQDN is basically a DNS host name and it represents where to resolve this host name within the DNS hierarchy.

The Domain Name Space:

As a tree is traversed in an ascending manner (i.e., from the leaf nodes to the root), the nodes become increasingly less specific (i.e., the leftmost label is most specific and the right most label is least specific). Typically in an FQDN, the left most label is the host name, while the next label to the right is the local domain to which the host belongs. The local domain can be a subdomain of another domain. The name of the parent domain is then the next label to the right of the subdomain (i.e., local domain) name label, and so on, till the root of the tree is reached.

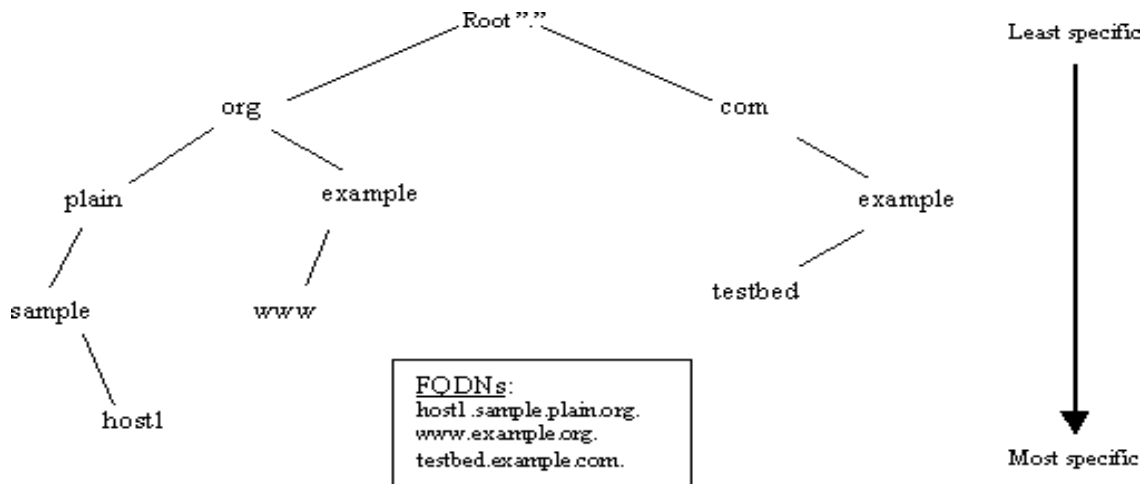


Figure 1:Domain name space example

The DNS is a hierarchical tree structure whose root node is known as the root domain. A label in a DNS name directly corresponds with a node in the DNS tree structure. A label is an alphanumeric string that uniquely identifies that node from its brothers. Labels are connected together with a dot notation, ".", and a DNS name containing multiple labels represents its path along the tree to the root. Labels are written from left to right. Only one zero length label is allowed and is reserved for the root of the tree. This is commonly referred to as the root zone. Due to the root label being zero length, all FQDNs end in a dot [RFC 1034].

When the DNS is used to map an IP address back into a host name (i.e., inverse resolution), the DNS makes use of the same notion of labels from left to right (i.e., most specific to least specific) when writing the IP address. This is in contrast to the typical representation of an IP address whose dotted decimal notation from left to right is least specific to most specific.

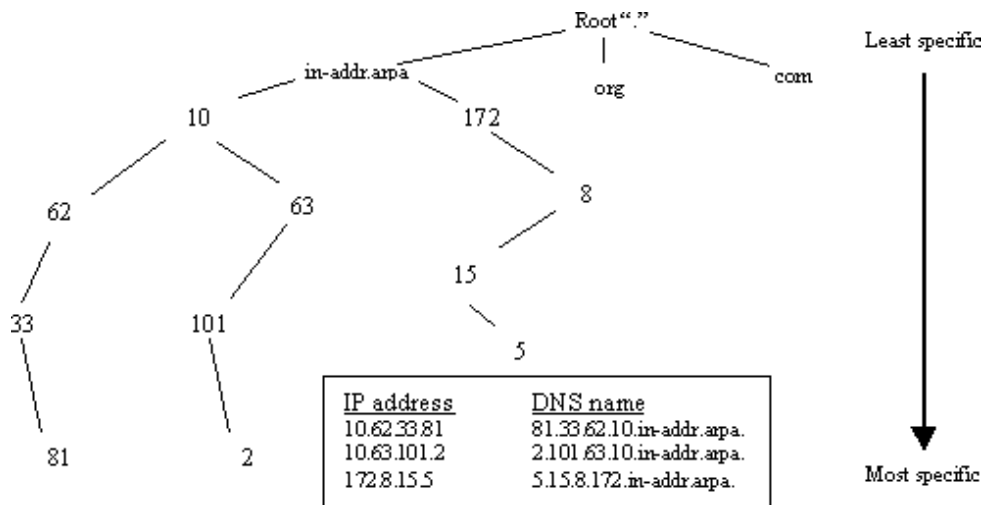


Figure 2: Example of inverse domains and the Domain Name Space

To handle this, IP addresses in the DNS are typically represented in reverse order. IP addresses fall under a special DNS top level domain (TLD), known as the in-addr.arpa domain. By doing this, using IP addresses to find DNS host names are handled just like DNS host name lookups to find IP addresses.

DNS Components

The DNS has three major components, the database, the server, and the client [RFC 1034]. The database is a distributed database and is comprised of the Domain Name Space, which is essentially the DNS tree, and the Resource Records (RRs) that define the domain names within the Domain Name Space. The server is commonly referred to as a name server. Name servers are typically responsible for managing some portion of the Domain Name Space and for assisting clients in finding information within the DNS tree. Name servers are authoritative for the domains in which they are responsible. They can also serve as a delegation point to identify other name servers that have authority over subdomains within a given domain.

The RR data found on the name server that makes up a domain is commonly referred to as zone information. Thus, name servers have zones of authority. A single zone can either be a forward zone (i.e., zone information that pertains to a given domain) or an inverse zone (i.e., zone information that maps IP addresses into DNS host names). DNS allows more than one name server per zone, but only one name server can be the primary server for the zone. Primary servers are where the actual changes to the data for a zone take place. All the other name servers for a zone basically maintain copies of the primary server's database for the zone. These servers are commonly referred to as secondary servers.

A DNS RR has 6 fields: NAME, TYPE, CLASS, TTL, RD Length, and RDATA. The NAME field holds the DNS name, also referred to as the owner name, to which the RR belongs. The TYPE field is the TYPE of RR. This field is necessary because it is not uncommon for a DNS name to have more than one type of RR.

RECORD TYPE	DESCRIPTION	USAGE
A	An address record	Maps FQDN into an IP address
PTR	A pointer record	Maps an IP address into FQDN
NS	A name server record	Denotes a name server for a zone
SOA	A Start of Authority record	Specifies many attributes concerning the zone, such as the name of the domain (forward or inverse), administrative contact, the serial number of the zone, refresh interval, retry interval, etc.
CNAME	A canonical name record	Defines an alias name and maps it to the absolute (canonical) name
MX	A Mail Exchanger record	Used to redirect email for a given domain or host to another host

Table 3

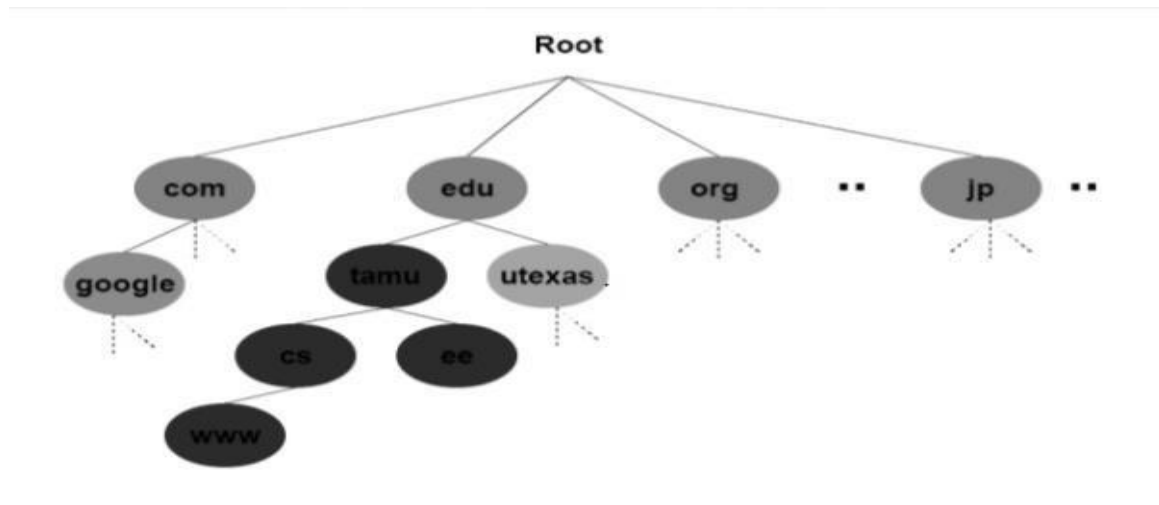


Figure 3: ARCHITECTURE OF DNS:

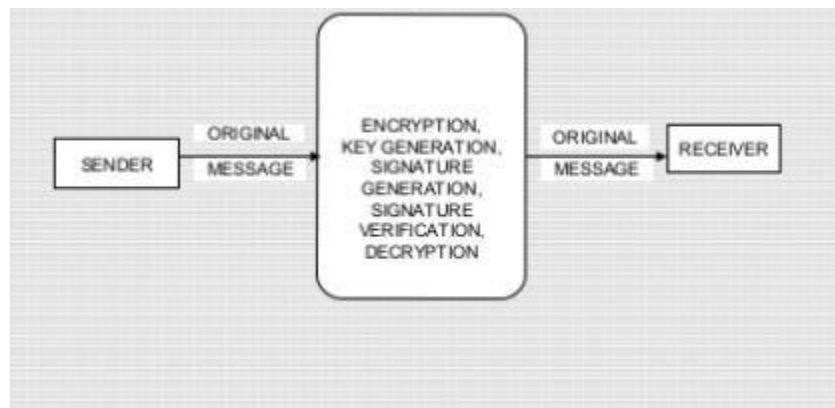


Figure 4: OVERALL DIAGRAM

6.

MODULE DESCRIPTION:

KEY GENERATION MODULE DESIGN:

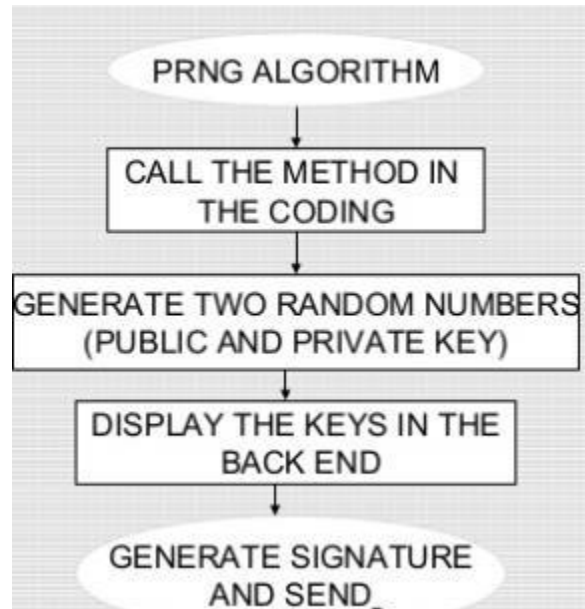


Figure 5:KEY GENERATION

AUTHENTICATION

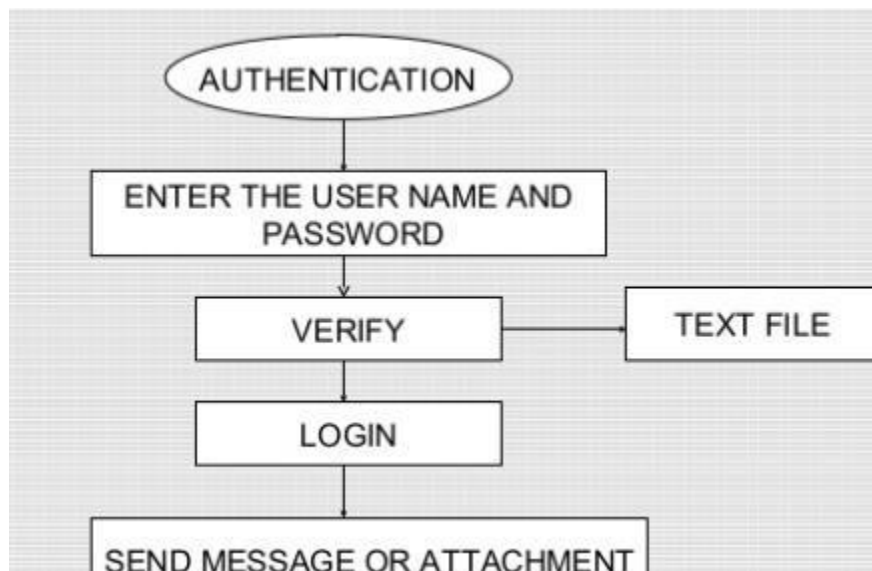


Figure 6:Authentication

MESSAGE ENCRYPTION:

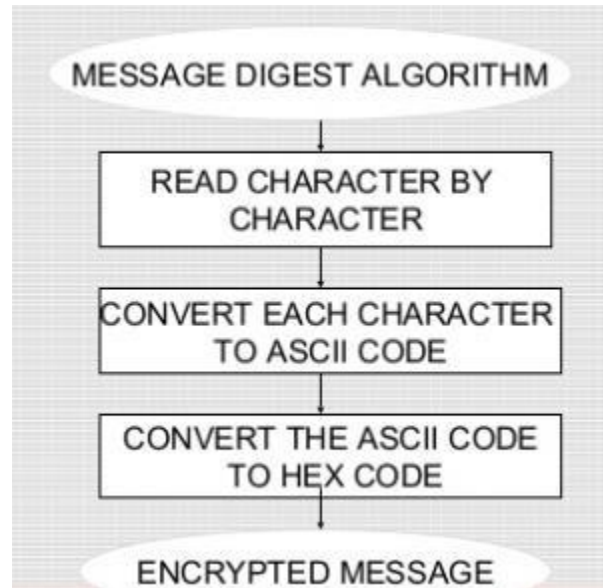


Figure 7:message encryption

SIGNATURE GENERATION:

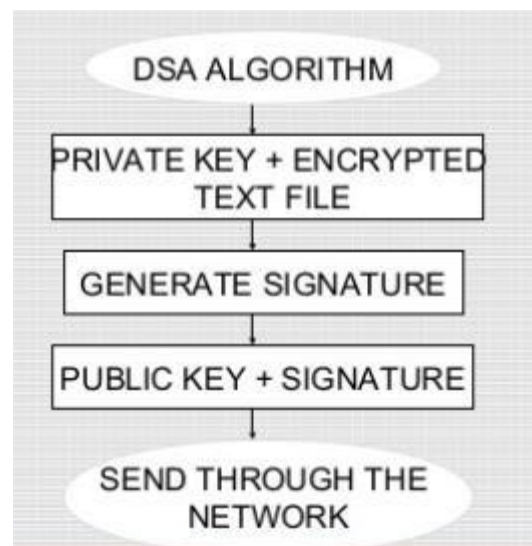


Figure 8:signature generation

6.5 DECRYPTION:

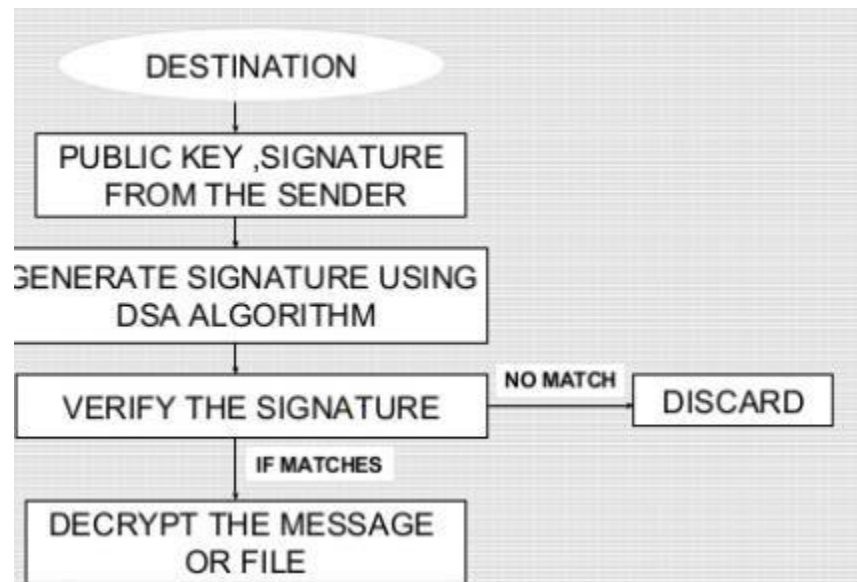


Figure 9:verifying signature and decryption

7. IMPLEMENTATION

:

7.1 CODING

```
import java.io.*;                                swing1.java
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;

public class Swing1 implements ActionListener
{
    JFrame f;
    JButton Submit,Reset;
    JTextField t1;
    JPasswordField t2;

    Swing1()
    {
        String inf="com.sun.java.swing.plaf.windows.WindowsLookAndFeel";
        try
        {
            UIManager.setLookAndFeel(inf);
        }
        catch(Exception e){}
    }

    public void Met()
```

```
{

JLabel l1 = new JLabel("UserName :",JLabel.RIGHT);
JLabel l2 = new JLabel("PassWord :",JLabel.RIGHT);
t1 = new JTextField(10);
t2 = new JPasswordField(10);
l1.setLabelFor(t1);
l2.setLabelFor(t2);
Submit = new JButton("Submit");
Reset = new JButton("Reset");

JPanel p = new JPanel();
p.setLayout(new GridLayout(3,2,5,5));
p.add(l1);
p.add(t1);
p.add(l2);
p.add(t2);
p.add(Submit);
p.add(Reset);
Submit.addActionListener(this);
Reset.addActionListener(this);
f = new JFrame("Login");
f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
f.setContentPane(p);
f.setBounds(250,250,180,120);
```

```

//f.pack();

f.setResizable(false);

f.setVisible(true);

}


public void actionPerformed(ActionEvent e)

{


if(Submit==(JButton) e.getSource())

{

if(((t1.getText()).equals("admin")) && ((t2.getText()).equals ("admin")))

{

Client c1 = new Client();

c1.setVisible(true);

f.setVisible(false);

}

else

{

JOptionPane.showMessageDialog(null, "Invaild UserName and PassWord","Alert",
JOptionPane.ERROR_MESSAGE);

t1.setText("");

t2.setText("");

f.setVisible(true);

}

}

}

```

```

else if(Reset == (JButton)e.getSource())
{
t1.setText("");
t2.setText("");
}
}

public static void main(String arg[])throws Exception
{
Swing1 s1 = new Swing1();
s1.Met();
}
}

```

DnsServer12.java

```

import java.io.*;
import java.net.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

class DnsServer12 implements ActionListener

{

static String str2,str21,str31,str41,str51,str71,str72;
static String str80,str81,str82,str83,str84,str85;

```

```

static JFrame jf;

static JTextField t1,t2,t3,t4,t5,t6;

static JLabel l,l1,l2,l3,l4,l5,l6,l7;

static JTextArea ta;

static JScrollPane js;

static JButton Send,Exit;

static JPanel p;

static Socket s2;

static String subs1;

static String sub="";

static String output;

static String output1;

// boolean succ;

DnsServer12()
{
String inf="com.sun.java.swing.plaf.windows.WindowsLookAndFeel";

try
{
    UIManager.setLookAndFeel(inf);
}

catch(Exception e){ }

jf =new JFrame("DNSServer");

l = new JLabel("DNS-Server Contains the Domains");

l1 = new JLabel("SenderName          :");

l2 = new JLabel("ReceiverName        :");

```

```

l3 = new JLabel("Public Key      :");
l4 = new JLabel("Signature      :");
l5 = new JLabel("EncyyptedData  :");
l6 = new JLabel("SenderDomain   :");
l7 = new JLabel("ReceiverDomain  :");

t1 = new JTextField(20);
t2 = new JTextField(20);
t3 = new JTextField(20);
t4 = new JTextField(20);
t5 = new JTextField(20);
t6 = new JTextField(20);

ta = new JTextArea();
js = new JScrollPane(ta);

Send= new JButton("Send");
Exit= new JButton("Exit");

Font f = new Font("Bold Italic",Font.PLAIN,20);

l.setFont(f);

p = new JPanel();

p.add(l);
p.add(l1);
p.add(l2);
p.add(l3);
p.add(l4);
p.add(l5);
p.add(l6);

```

```
p.add(l7);  
p.add(t1);  
p.add(t2);  
p.add(t3);  
p.add(t4);  
p.add(t5);  
p.add(t6);  
p.add(js);  
p.add(Send);  
p.add(Exit);  
Send.addActionListener(this);  
Exit.addActionListener(this);  
p.setLayout(null);  
l.setBounds(40,40,350,25);  
l1.setBounds(40,70,120,25);  
t1.setBounds(180,70,120,25);  
l2.setBounds(40,100,120,25);  
t2.setBounds(180,100,120,25);  
l3.setBounds(40,130,120,25);  
t3.setBounds(180,130,120,25);  
l4.setBounds(40,160,120,25);  
t4.setBounds(180,160,120,25);  
l6.setBounds(40,190,120,25);  
t5.setBounds(180,190,120,25);  
l7.setBounds(40,220,120,25);
```



```

t6.setBounds(180,220,120,25);
l5.setBounds(40,250,120,25);
js.setBounds(40,270,300,180);
Send.setBounds(100,470,75,25);
Exit.setBounds(200,470,75,25);
jf.setContentPane(p);
jf.setSize(400,530);
jf.setVisible(true);
}

public void actionPerformed(ActionEvent e12)
{
if(e12.getSource()==Send)
{

Soc1();
//d1.ServSoc1();
//d1.Soc1();
}
else
{
System.out.println("Connection Refused");
}

if(Exit== e12.getSource())

```

```
{  
    System.exit(0);  
}
```

```
}
```

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

```
{
```

```
    DnsServer12 DS = new DnsServer12();
```

```
    DS.ServSoc1();
```

```
    //DS.Soc1();
```

```
    //DS.ServSoc11();
```

```
    //DS.Soc11();
```

```
}
```

```
public static void ServSoc1()
```

```
{
```

```
    try
```

```
    {
```

```
        ServerSocket ss = new ServerSocket(9999);
```

```
        Socket s1 = ss.accept();
```

```
BufferedReader ps2=new BufferedReader(new  
InputStreamReader(s1.getInputStream()));
```

```
str2 = ps2.readLine();
```

```
str2=str2.trim();
```

```
System.out.println("SenderDomain :"+str2);
```

```
str21 = ps2.readLine();
```

```
str21=str21.trim();
```

```
System.out.println("ReceiverDomain:"+str21);
```

```
str31 = ps2.readLine();
```

```
str31=str31.trim();
```

```
System.out.println("SenderName :"+str31);
```

```
str41 = ps2.readLine();
```

```
str41=str41.trim();
```

```
System.out.println("ReceiverName :"+str41);
```

```
str51 = ps2.readLine();
```

```
str51=str51.trim();
```

```
System.out.println("EncryptedData :"+str51);
```

```
try
```

```
{
```

```
int len=str51.length();
```

```

System.out.println("Length:"+len);

int len1=len/8;

System.out.println("Length1:"+len1);

int len2=len%8;

int k=8;

System.out.println("Length2:"+len2);

int c=0;

for(int l=0;l<len;l+=8)

{

System.out.println("Source:"+str31);

if(k<=len-len2)

{

String subs = str51.substring(l,k);

subs1 = subs + "\n";

sub+=subs1;

System.out.println("Stirnfg1:"+subs1);

c++;

k+=8;

System.out.println("inside "+k);

}

System.out.println("Destination:"+str41);

}

if(len2>0)

{

```

```

System.out.println("Source:"+str31);
System.out.println("inside ");
int g =(len- c*8);
System.out.println("Value:"+g);
subs1 = str51.substring(c*8,len);
sub+=subs1;
System.out.println("String2:"+subs1);
System.out.println("Destination:"+str41);
}
ta.setText("Source :"+str31+"\n"+sub+"\n"+"Destination :"+str41);

}

catch(Exception e)
{
e.printStackTrace();
}


str71 = ps2.readLine();
String sstr = str71;
str71=str71.trim();
System.out.println("Public Key  :"+str71);


str72 = sstr;
str72=str72.trim();

```

```

System.out.println("Signature    :"+str72);

System.out.println("Server Checking the List of Domains");

System.out.println("Now Checking wih Domain1");

Dom1(str2);

System.out.println("Now Checking wih Domain2");

Dom2(str21);

s1.close();

}

catch(Exception e)

{

System.out.println(e);

}

}

public static void Soc1()

{

try

{

System.out.println("I am inside try");

/*if(output.equals("Service Protected"))

{

JOptionPane.showMessageDialog(null,"Connection
Terminated","Connection",JOptionPane.ERROR_MESSAGE);

s2.close();

}

```

```

else if(output1.equals("Service Protected"))
{
JOptionPane.showMessageDialog(null,"Connection
Terminated","Connection",JOptionPane.ERROR_MESSAGE);

s2.close();

}

else*/

{

String inputValue = JOptionPane.showInputDialog("Enter the System Name for
Domain2");

Socket s2 = new Socket(InetAddress.getByName(inputValue),7878);

System.out.println("Socket Created");

PrintStream ps = new PrintStream(s2.getOutputStream());

///

System.out.println("str2 " + str2);
System.out.println("str21 " + str21);
System.out.println("str31 " + str31);
System.out.println("str41 " + str41);
System.out.println("str51 " + str51);
System.out.println("str71 " + str71);
System.out.println("str72 " + str72);

ps.println(str2);

ps.println(str21);

ps.println(str31);

ps.println(str41);

ps.println(str51);

```

```

ps.println(str71);
ps.println(str72);
ps.flush();
System.out.println("Data Has Been written");
s2.close();
}

}
catch(Exception e)
{
System.out.println("i am printing " + e);
}
}

public static void Dom1(String temp)throws Exception

{
int Result;
String Response;
String Rname;
Result=0;
Rname=temp;
FileReader fr = new FileReader("c:\\file1.txt");
BufferedReader br = new BufferedReader(fr);
System.out.println("File has readed sucessfully");

```



```
String s="";  
String str[];  
  
while((s=br.readLine())!=null)  
{  
  
str=s.split(" ");  
  
if(Rname.equalsIgnoreCase(str[0]))  
{  
  
Result=1;  
output=str[1];  
System.out.println("SenderDomainIP : "+output);  
break;  
  
}  
  
else  
{  
output="Service Protected";  
System.out.println("SenderDomainIP : "+output);  
}  
}
```

```
}
```

```
if(output.equals("Service Protected"))
```

```
{
```

```
JOptionPane.showMessageDialog(null,"Server  
Protected","Service",JOptionPane.ERROR_MESSAGE);
```

```
}
```

```
else
```

```
{
```

```
JOptionPane.showMessageDialog(null,"Server  
Provided","Service",JOptionPane.INFORMATION_MESSAGE);
```

```
t1.setText(str31);
```

```
t5.setText(str2);
```

```
}
```

```
}
```

```
public static void Dom2(String temp)throws Exception
```

```
{
```

```
int Result;
```

```
String Response;
```

```
String Rname;
```

```
Result=0;
```

```
Rname=temp;
```

```
FileReader fr = new FileReader("c:\\file1.txt");
```

```
BufferedReader br = new BufferedReader(fr);

String s="";

String str[];

while((s=br.readLine())!=null)
{

str=s.split(" ");

if(Rname.equalsIgnoreCase(str[0]))
{

Result=1;

output1=str[1];

System.out.println("ReceiverDomainIP : "+output1);

break;

}

else
{

output1="Service Protected";

System.out.println("ReceiverDomainIP : "+output1);
```

```

    }

    }

    if(output1.equals("Service Protected"))

    {

        JOptionPane.showMessageDialog(null,"Server
        Protected","Service",JOptionPane.ERROR_MESSAGE);

    }

    else

    {

        JOptionPane.showMessageDialog(null,"Server
        Provided","Service",JOptionPane.INFORMATION_MESSAGE);

        t6.setText(str21);
        t2.setText(str41);
        t3.setText(str71);
        t4.setText(str72);

    }

    }

}

```

SwingRMes.java

```

import java.io.*;

import java.awt.*;

import java.awt.event.*;

```

```

import java.util.*;

import java.net.*;

import java.util.zip.*;

import java.security.*;

import java.security.SecureRandom.*;

import java.security.spec.*;

import javax.swing.*;

public class SwingRMes extends JFrame implements ActionListener
{
    JTabbedPane tp;

    JLabel jl,jl1,jl2,jl;

    JButton jb,jb1,jb2;

    JTextField tf1,tf2,tfd,tfd1;

    JTextArea tf,ta1;

    JScrollPane sp,sp1;

    static String g1=" ";

    static String n,n1,n2;

    static byte str[];

    static byte realSig[];

    static byte realSig1[];

    static String y="";

    static String str6,str61,str2,str12,str13;

    static String t,st="",st1="",vj="";

```

```

public SwingRMes()
{
    super("Reception");
    String inf="com.sun.java.swing.plaf.windows.WindowsLookAndFeel";
    try
    {
        UIManager.setLookAndFeel(inf);
    }
    catch(Exception e){ }

    setSize(400,450);

    Container c = getContentPane();

    JPanel jp = new JPanel();
    jp.setLayout(null);
    jl = new JLabel("Sender Name ",JLabel.LEFT);
    jL = new JLabel("Receiver Name ",JLabel.LEFT);
    tfd = new JTextField(15);
    tfd1 = new JTextField(15);
    tf = new JTextArea(20,40);
    sp = new JScrollPane(tf);
    jb= new JButton("OPEN");
    jp.add(jl);
    jp.add(tfd);
    jp.add(jL);

```

```
jp.add(tfd1);
```

```
jp.add(sp);
```

```
jp.add(jb);
```

```
jl.setBounds(25,10,150,25);
```

```
tfd.setBounds(150,10,150,25);
```

```
jL.setBounds(25,40,150,25);
```

```
tfd1.setBounds(150,40,150,25);
```

```
sp.setBounds(25,70,300,250);
```

```
jb.setBounds(150,350,75,25);
```

```
jp.setVisible(true);
```

```
JPanel jp1 = new JPanel();
```

```
jp1.setLayout(null);
```

```
j11 = new JLabel("Sender Name ",JLabel.LEFT);
```

```
j12 = new JLabel("Receiver Name ",JLabel.LEFT);
```

```
tf1 = new JTextField(15);
```

```
tf2 = new JTextField(15);
```

```
ta1 = new JTextArea(20,40);
```

```
sp1 = new JScrollPane(ta1);
```

```
jb2= new JButton("Retrive");
```

```
j11.setBounds(25,10,150,25);
```

```
tf1.setBounds(150,10,150,25);
```

```
j12.setBounds(25,40,150,25);
```

```
tf2.setBounds(150,40,150,25);
```

```

sp1.setBounds(25,70,300,250);
jb2.setBounds(150,350,75,25);
jb.addActionListener(this);
jb2.addActionListener(this);
jp1.add(jl1);
jp1.add(tf1);
jp1.add(jl2);
jp1.add(tf2);
jp1.add(sp1);
jp1.add(jb2);
jp1.setVisible(true);

tp = new JTabbedPane();
tp.addTab("Message",null,jp,"Retrive the Message here");
tp.addTab("File",null,jp1,"Retrive the File Name here");
c.add(tp);

setDefaultCloseOperation(EXIT_ON_CLOSE);
setVisible(true);
setResizable(false);
}

public static void main(String arg[])throws Exception
{
SwingRMes rm = new SwingRMes();
rm.ServSoc();

```



```
}
```

```
public static void ServSoc()
```

```
{
```

```
try
```

```
{
```

```
System.out.println(" Destination Server Started.....");
```

```
ServerSocket ss = new ServerSocket(7878);
```

```
System.out.println(" Socket Created. ....");
```

```
Socket s11 = ss.accept();
```

```
BufferedReader ps2=new BufferedReader(new InputStreamReader(s11.getInputStream()));
```

```
str6 = ps2.readLine();
```

```
str6=str6.trim();
```

```
System.out.println("SenderName : " +str6);
```

```
str61 = ps2.readLine();
```

```
str61=str61.trim();
```

```
System.out.println("ReceiverName : " +str61);
```

```
str2 = ps2.readLine();
```

```
str2=str2.trim();
```

```
System.out.println("File Name   : " +str2);
```

```
str12 = ps2.readLine();
```

```
str12=str12.trim();
```

```
System.out.println("Public Key : " +str12);
```

```
str13 = ps2.readLine();
```

```

str13=str13.trim();

System.out.println("RealSign    : " +str13);


char a[]=new char[50000];

char res[]=new char[9];

char s1[]=new char[5];

String w =new String();

String r =new String();

int i,j=0,d=1,h,n,l=8,l2=0,l1=0;

long c=0l;

String t,st="",st1="";


/* Getting the Hexcode And Written into TextFile */

System.out.println("File Name to be Opened" + str13);

InputStream in3 = new FileInputStream(str13);


System.out.println("Trying to out stream");

FileOutputStream out2=new FileOutputStream("Sign1.txt");

byte[] str3=new byte[512];

while(in3.read(str3)!=-1)

{

String r1=new String(str3);

out2.write(str3);

```

```
}
```

```
System.out.println("Enter the Hex-Code:");
```

```
System.out.println(new String(str12));
```

```
int len=str12.length();
```

```
System.out.print("String Length:");
```

```
System.out.println(len);
```

```
///changed 2 ad 12 /* Converting Hexcode into Bits */
```

```
for(i=0;i<len;i++)
```

```
{
```

```
char aa= ((str12).charAt(i));
```

```
a[i]=aa;
```

```
if(a[i]=='A')
```

```
{
```

```
st="1010";
```

```
st1=st;
```

```
}
```

```
else if(a[i]=='B')
```

```
{
```

```
st="1011";
```

```
st1=st;
```

```
}
```

```
else if(a[i]=='C')
```

```
{
```

```
st="1100";  
st1=st;  
}  
else if(a[i]=='D')  
{  
st="1101";  
st1=st;  
}  
else if(a[i]=='E')  
{  
st="1110";  
st1=st;  
}  
else if(a[i]=='F')  
{  
st="1111";  
st1=st;  
}  
else if(a[i]=='0')  
{  
st="0000";  
st1=st;  
}  
else if(a[i]=='1')  
{  
st="0001";
```

```
    st1=st;
}
else if(a[i]=='2')
{
    st="0010";
    st1=st;
}
else if(a[i]=='3')
{
    st="0011";
    st1=st;
}
else if(a[i]=='4')
{
    st="0100";
    st1=st;
}
else if(a[i]=='5')
{
    st="0101";
    st1=st;
}
else if(a[i]=='6')
{
    st="0110";
    st1=st;
```

```

    }

    else if(a[i]=='7')

    {

    st="0111";

    st1=st;

    }

    else if(a[i]=='8')

    {

    st="1000";

    }

    else if(a[i]=='9')

    {

    st="1001";

    }

    w=w+st;

    }

    char w_ch[]=w.toCharArray();

    String x=w.toString();

    System.out.print("this is content " + x);

    System.out.println("\nLength:"+len);


    /* Converting Hex into Character */

    ///

    try

    {

    if(len==8)

```

```

{
int k;
k=len;
for(j=k-1;j>=0;j--)
{
c=c+((w_ch[j])-48)*d;
y=y+c;
d=d*2;
}
System.out.print("\nCharacter after Decryption :");
vj=vj+(char)c;

}
else
{
int k;
for(i=0;i<50;i++)
{
c=0; d=1;
k=i*8;
for(j=k+7;j>=k;j--)
{
c=c+(w_ch[j]-48)*d;
y=y+c;
d=d*2;

```

```

    }
    vj=vj+(char)c;
    System.out.println(vj);
    }

    }

    }
    catch(Exception e)
    {
        System.out.print("\ni am printing " + e);

    }
    FileOutputStream fos=new FileOutputStream("Recv.txt");
    fos.write(vj.getBytes());
    s11.close();

    }
    catch(Exception e)
    {

    }
    }

    public static boolean verifySig(String fname) throws Exception

```



```

{

byte[] md;

byte[] sign;

byte[] realSig;

String args=fname;

FileInputStream fin=new FileInputStream(args);

byte[] in_text=new byte[fin.available()];

fin.read(in_text);

fin.close();

//SHA sha=new SHA(in_text);

md=digestValue(in_text);


FileInputStream finPublic=new FileInputStream(str12);

byte[] enc_pub=new byte[finPublic.available()];

finPublic.read(enc_pub);

finPublic.close();


FileInputStream sigfis=new FileInputStream(str13);

sign=new byte[sigfis.available()];

sigfis.read(sign);


Signature dsa1 = Signature.getInstance("SHA1withDSA","SUN");

X509EncodedKeySpec pubKeySpec=new X509EncodedKeySpec(enc_pub);

KeyFactory keyFactoryPub=KeyFactory.getInstance("DSA","SUN");

```

```
PublicKey pub=keyFactoryPub.generatePublic(pubKeySpec);  
dsa1.initVerify(pub);  
dsa1.update(md);  
boolean status=dsa1.verify(sign);  
System.out.println("Verified Signature:" + status);  
return status;  
}
```

```
private static byte[] content;
```

```
public static byte[] digestValue(byte[] in_text1)
```

```
{  
    byte[] in_text=in_text1;  
    content=in_text;  
    MessageDigest mg1=null;  
    try  
    {  
        mg1=MessageDigest.getInstance("SHA1");  
    }
```

```
    catch (Exception e)  
    {  
        System.out.println(e);  
    }
```

```
    mg1.update(content);
```

```

byte[] digest1=mg1.digest();

System.out.println("Message Digest:"+digest1);

return digest1;

}


public void actionPerformed(ActionEvent e)

{

if(e.getSource() == jb2)

{

JOptionPane.showMessageDialog(null, "Signature are Verified", "Verify Sign",
JOptionPane.INFORMATION_MESSAGE);

JOptionPane.showMessageDialog(null, "Receiver accept the data from
Domain2", "Receiver", JOptionPane.INFORMATION_MESSAGE);

tf1.setText(str6);

tf2.setText(str61);

ta1.setText(vj);

}

else if(e.getSource() == jb)

{

System.out.println("Verify Signature are Generated");


try

{

verifySig(str12);

}

catch(Exception e1)

{

```

```
System.out.println(e1);
```

```
}
```

```
JOptionPane.showMessageDialog(null, "Signature are Verified", "Verify Sign",  
JOptionPane.INFORMATION_MESSAGE);
```

```
JOptionPane.showMessageDialog(null, "Receiver accept the data from  
Domain2", "Receiver", JOptionPane.INFORMATION_MESSAGE);
```

```
tfd.setText(str6);
```

```
tfd1.setText(str61);
```

```
//System.out.print("FileContent"+vj);
```

```
tf.setText(vj);
```

```
}
```

```
}
```

```
}
```

7.2 SCREENSHOTS:

Run swing1.java file

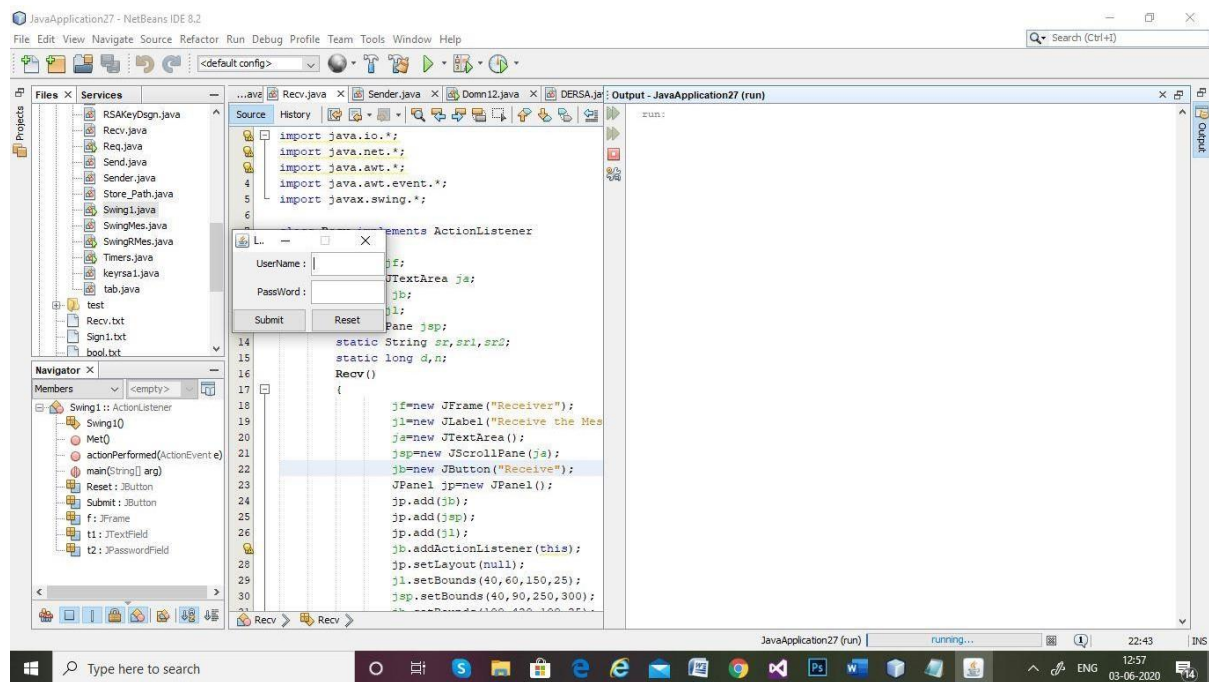
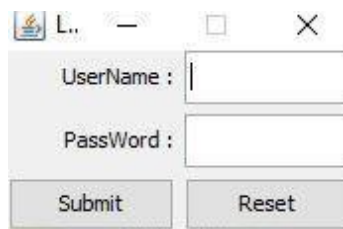


Figure 10:swing1.java

Now this screen will be displayed:



Now Run DnsServer12.java file:

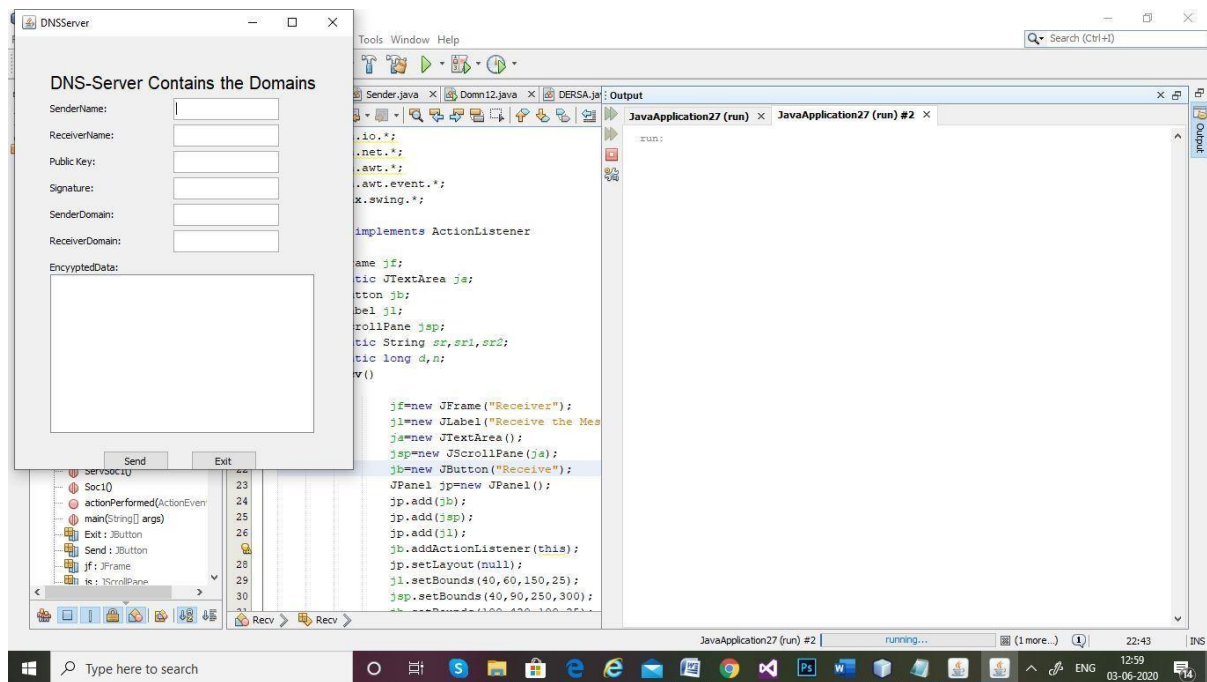


Figure 11: DnsServer12.java

Now run SwingRmes.java file

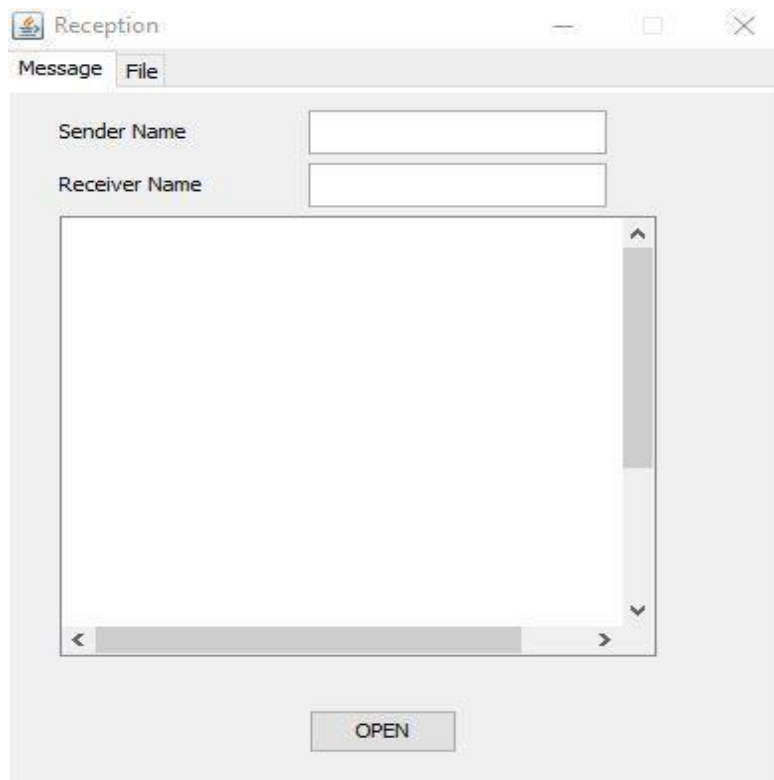
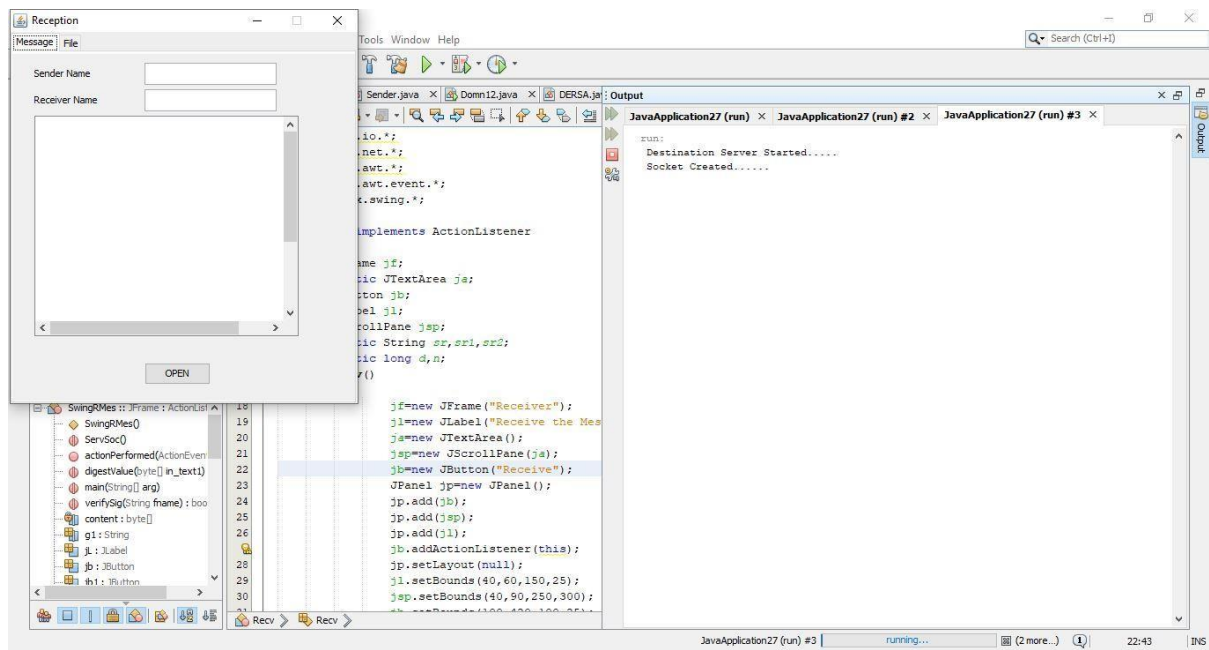


Figure 12: SwingRmes.java

Now open all the three screens parallelly and then enter the user id and password in the first screen:

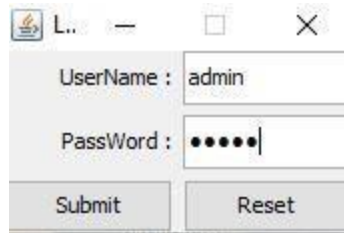
A small application window titled "L.." with standard Windows window controls (minimize, maximize, close). It contains two text input fields. The first is labeled "UserName :" and contains the text "admin". The second is labeled "PassWord :" and contains seven dots, indicating a masked password. Below the fields are two buttons: "Submit" and "Reset".

Figure 13: userid and password in screen 1

After entering this screen will be displayed :

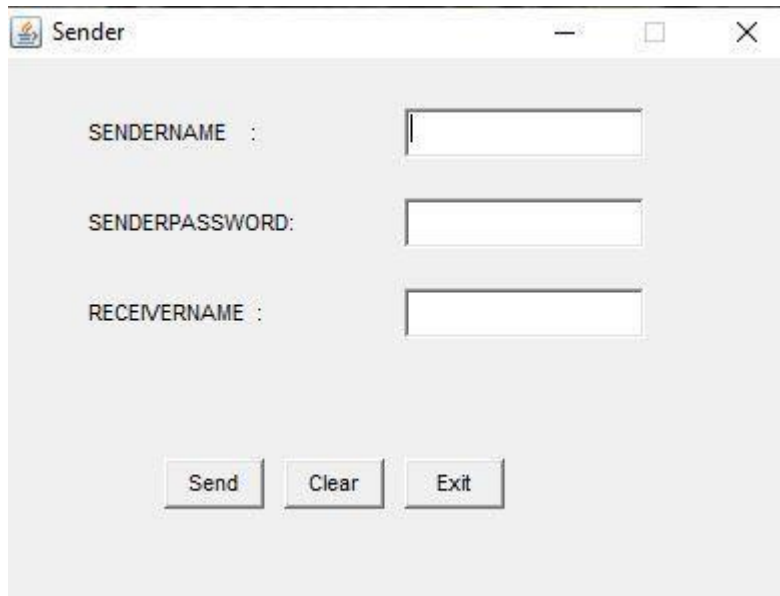
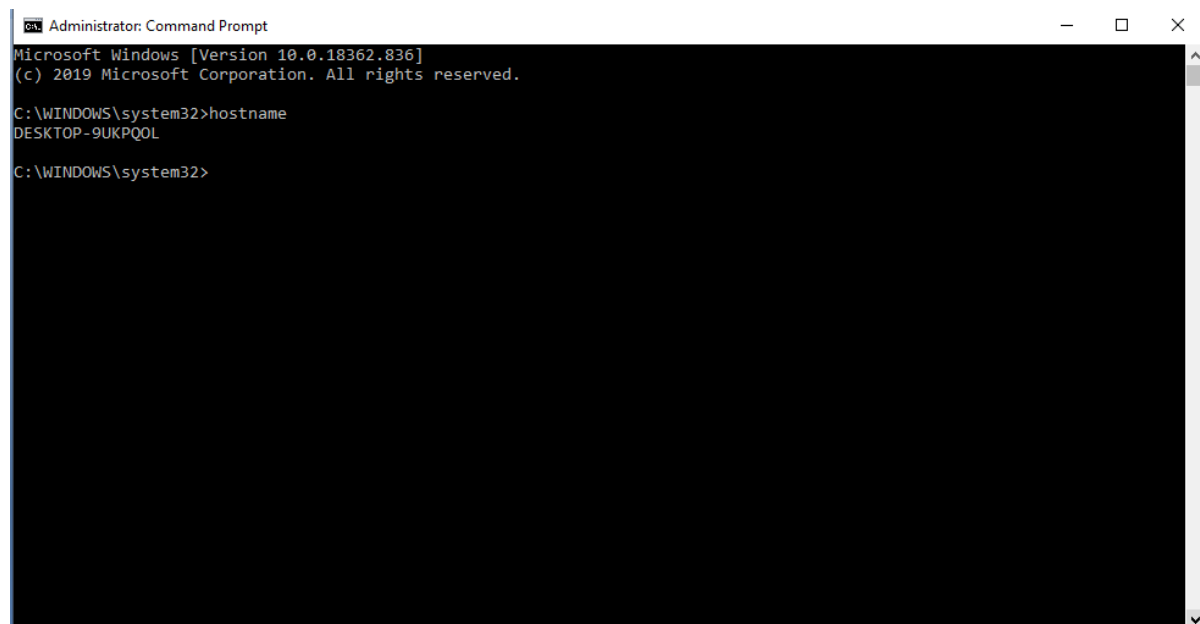
A window titled "Sender" with standard Windows window controls. It contains three text input fields. The first is labeled "SENDERNAME :", the second "SENDERPASSWORD:", and the third "RECEIVERNAME :". All three fields are currently empty. At the bottom of the window are three buttons: "Send", "Clear", and "Exit".

Figure 14: Sender screen.

We can get the sender name and the receiver name through the hostname command In the command prompt



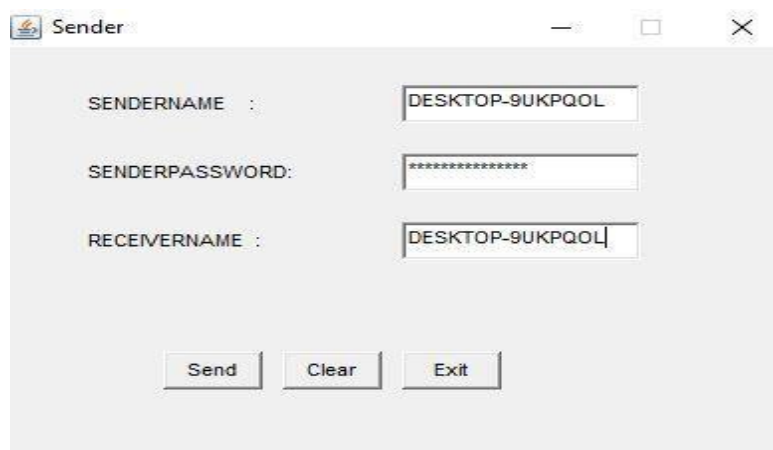
```
Administrator: Command Prompt
Microsoft Windows [Version 10.0.18362.836]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32>hostname
DESKTOP-9UKPQOL

C:\WINDOWS\system32>
```

Figure 15:domain name of the sytem

Now enter the name in the previous screen 7.2.6 in Sender name place and enter the receiver host name .Here we used the receiver name as same sender name so that I get the data to the same system if we need to get the data to a different system we can give the receiver name as the other system host name and then click on send button.



Sender

SENDERNAME :

SENDERPASSWORD:

RECEIVERNAME :

Figure 16 : Sender screen

Now on clicking send we get the below screen and now enter the text or send any file :

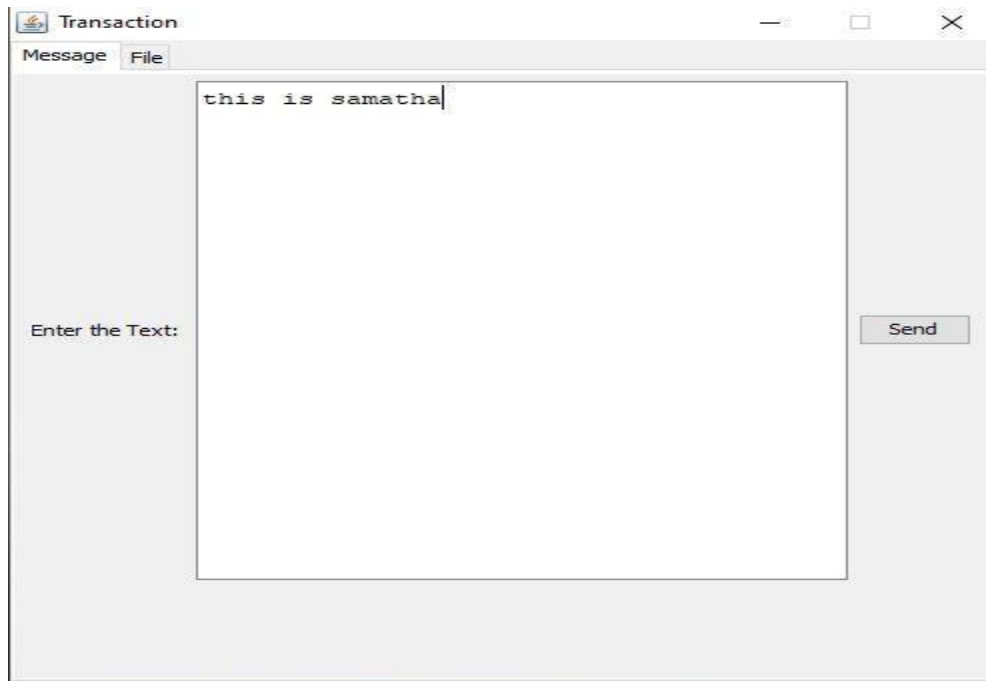


Figure 17: text entry screen

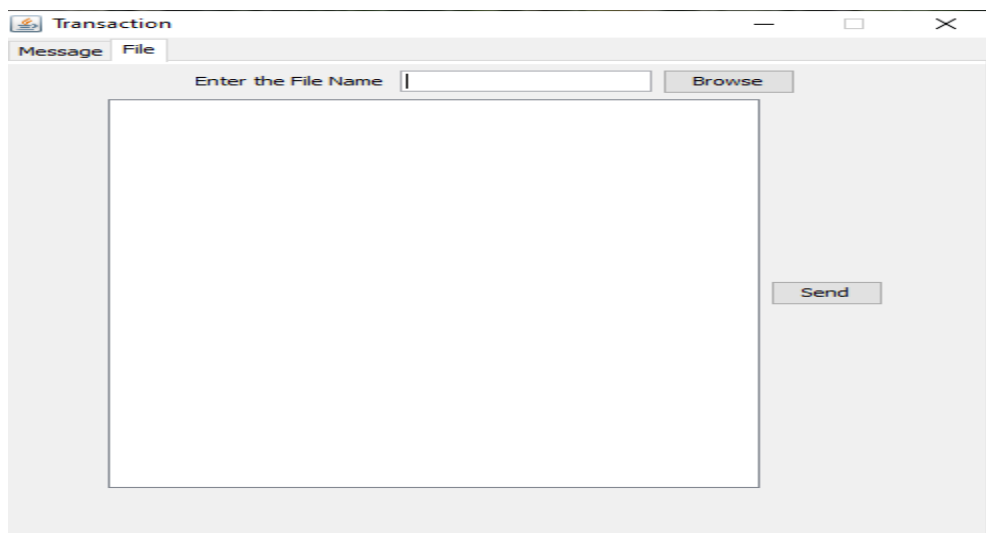


Figure 18

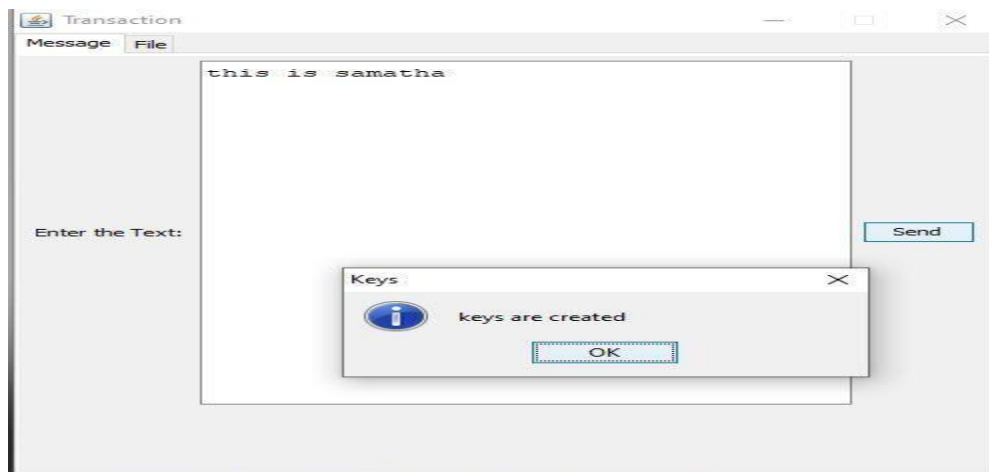


Figure 19: keys creation popup



Figure 20: Signature generation popup

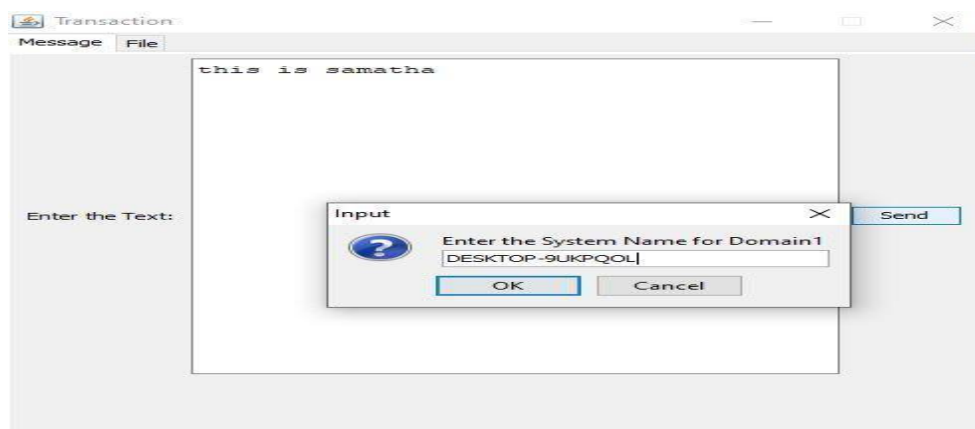


Figure 21: system name popup

Now after entering the domain name the Encrypted data gets generated and is displayed ayutomatically in the DNSserver screen as shown below

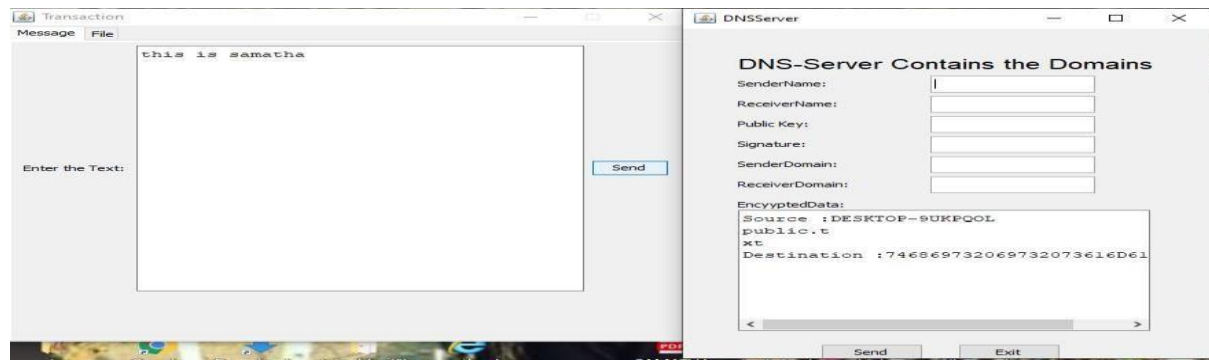


Figure 22: encrypted data

Now after that enter all the details in the DnsServer screen the public key and signature are displayed in the output screen and enter the data from that output screen.

```

Output
JavaApplication27 (run) X JavaApplication27 (debug) X JavaApplication27 (run) #2 X
Destination:746869732069732073616D61746861
Source:DESKTOP-9UKPQOL
Destination:746869732069732073616D61746861
Source:DESKTOP-9UKPQOL
inside
Value:2
String2:xt
Destination:746869732069732073616D61746861
Public Key :realSign.txt
Signature :realSign.txt
Server Checking the List of Domains
Now Checking with Domain1
java.io.FileNotFoundException: c:\file1.txt (The system cannot find the file specified)
I am inside try
Socket Created
str2 DESKTOP-9UKPQOL
str21 DESKTOP-9UKPQOL
str31 DESKTOP-9UKPQOL
str41 746869732069732073616D61746861
str51 public.txt
str71 realSign.txt
str72 realSign.txt
Data Has Been written
I am inside try
Socket Created
str2 DESKTOP-9UKPQOL
str21 DESKTOP-9UKPQOL
str31 DESKTOP-9UKPQOL
str41 746869732069732073616D61746861
str51 public.txt
str71 realSign.txt
str72 realSign.txt
Data Has Been written
BUILD SUCCESSFUL (total time: 9 minutes 18 seconds)

```

Now enter the details here in this screen and click ok

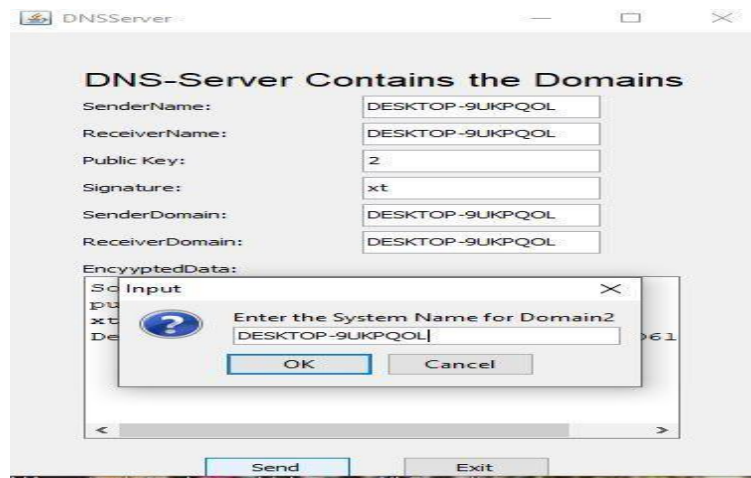


Figure 23 : Dns server screen

Now open the third screen SwingRmes.java file and enter the sender and receiver system name here I have given the same hostname for both sender and receiver so it is sent to the same system.and now it verifies the signature

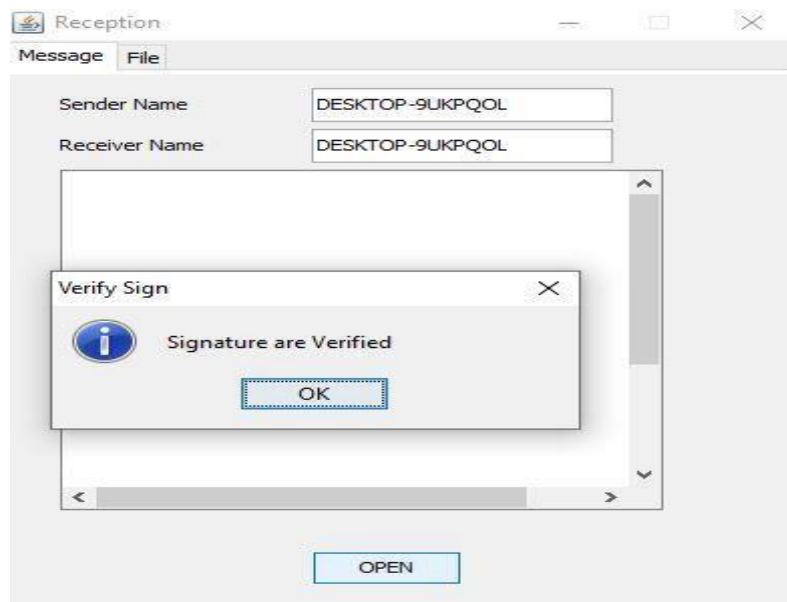
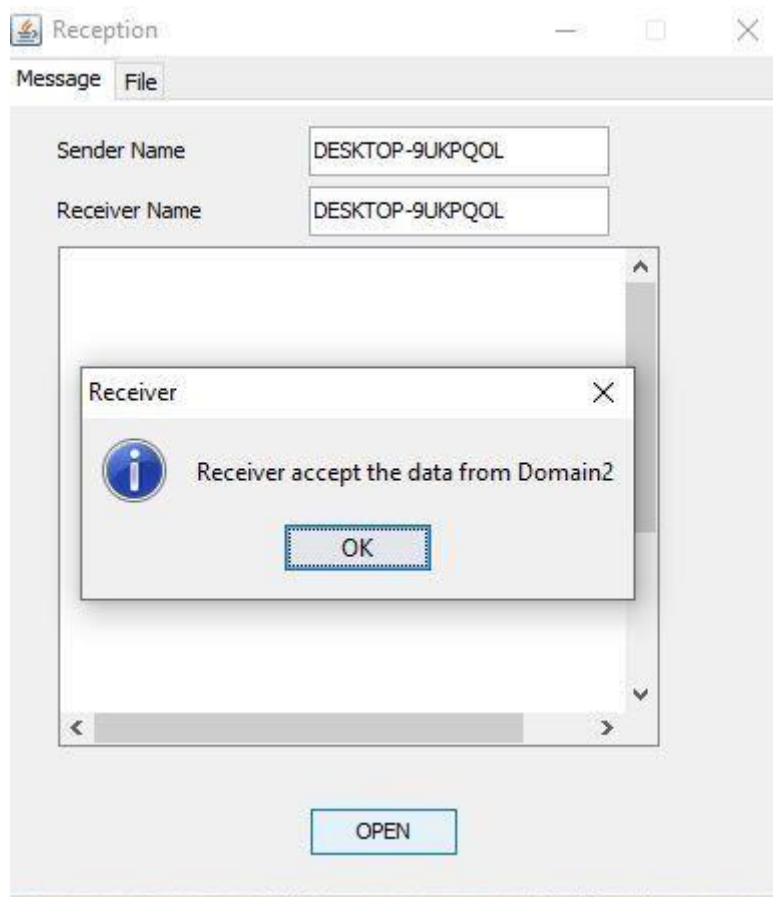


Figure 24 : signature verification



Now the data is sent to the reciever

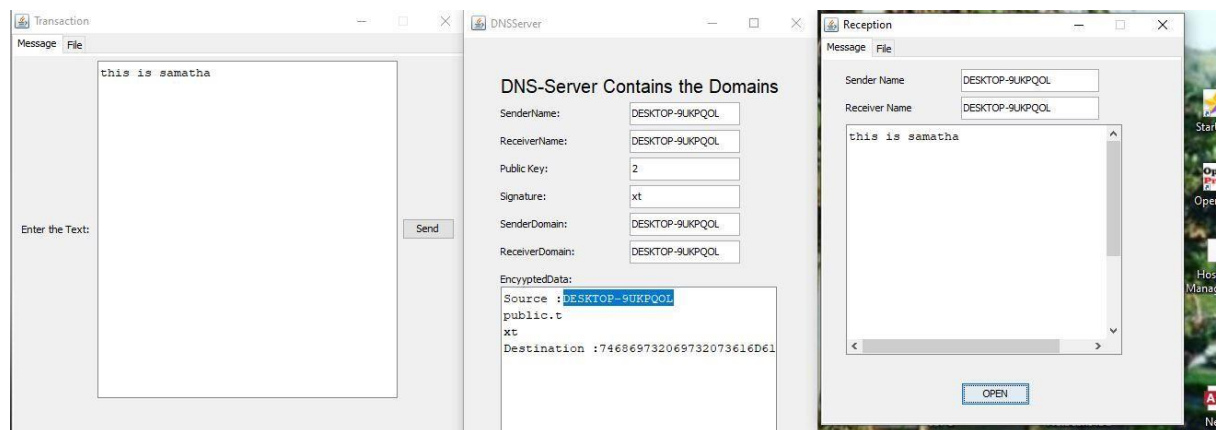


Figure 25: data is sent to the reciever

8. Conclusion

The Domain Name System (DNS) security working group (dnssec) will specify enhancements to the DNS protocol to protect the DNS against unauthorized modification of data and against masquerading of DNS data origin. That is, it will add data integrity and authentication capabilities to the DNS. The specific mechanism to be added to the DNS protocol will be a digital signature.

The digital signature service will be added such that the DNS resource records will be signed and, by distributing the signatures with the records, remote sites can verify the signatures and thus have confidence in the accuracy of the records received.

9. References

1. Albitz, P. and Liu, C., (1997) „DNS and Bind“, 2 nd Ed., Sebastopol, CA, O'Reilly & Associates, pp.1-9.
2. Herbert Schildt, Edition (2003) „The Complete Reference JAVA 2“ Tata McGraw Hill Publications
3. IETF DNSSEC WG, (1994) „DNS Security (dnssec) Charter“, IETF.
4. Michael Foley and Mark McCulley, Edition (2002) 'JFC Unleashed' Prentice-Hall India.
5. Mockapetris, P., (1987) „Domain Names - Concepts and Facilities“.