“Web Application for Secure File Transfer”

A

Project Report *submitted in final*

fulfillment of the requirements for

the award of the degree of

BACHELOR OF TECHNOLOGY in COMPUTER SCIENCE & ENGINEERING

by

Name	Roll No.
Shobhit Kumar Pal	**R134217154**
Shivam Gupta	**R134217151**
Teghpartap Singh Virk	**R134217192**
Sarthak Makkar	**R134217140**

under the guidance of

Mr.Akashdeep Bhardwaj

ASSISTANT PROFESSOR (SS)

School of Computer Science

Department of Systemics

University of Petroleum & Energy Studies

Bidholi, Via Prem Nagar, Dehradun, Uttarakhand

May – 2020

CANDIDATE'S DECLARATION

We hereby certify that the project work entitled **"Secure File Transfer Web Application"** in final fulfillment of the requirements for the award of the Degree of BACHELOR OF TECHNOLOGY in COMPUTER SCIENCE AND ENGINEERING with specialization in Cyber Security and Forensics and submitted to the School of Computer Science & Engineering at Center for Information Technology, University of Petroleum & Energy Studies, Dehradun, is an authentic record of our work carried out during a period from January, **2020** to **May**, **2020** under the supervision of **Mr Akashdeep Bhardwaj, Assistant Professor** .

The matter presented in this project has not been submitted by us for the award of any other degree of this or any other University.

Name	**Shobhit Kumar Pal**	**Shivam Gupta**	**TeghPartap Singh Virk**	**Sarthak Makkar**
Roll No.	**R134217154**	**R134217151**	**R134217192**	**R134217140**

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

Date: 13-05-2020

Mr.Akashdeep Bhardawaj
Project Guide

Dr.Neelu Jyoti Ahuja
HOD-Department of Systemics
School of Computer Science
University of Petroleum & EnergyStudies
Dehradun – 248001 (Uttarakhand)

ACKNOWLEDGEMENT

We wish to express our deep gratitude to our guide **Mr. Akashdeep Bhardawaj**, for all advice, encouragement and constant support he has given us through out our project work. This work would not have been possible without his support and valuable suggestions.

We sincerely thank to our respected Head of the Department, **Dr. Neelu Jyoti Ahuja**, for her great support in doing our project in **Cyber Security** at **SoCS**.

We are also grateful to **Dr. Manish Prateek** Dean SoCS, UPES for giving us the necessary facilities to carry out our project work successfully.

We would like to thank all our **friends** for their help and constructive criticism during our project work. Finally we have no words to express our sincere gratitude to our **parents** who have shown us this world and for every support they have given us.

Name	**Shobhit Kumar Pal**	**Shivam Gupta**	**Teghpartap Singh Virk**	**Sarthak Makkar**
Roll No.	**R134217154**	**R134217151**	**R134217192**	**R134217140**

ABSTRACT

Secure Data transfer based upon well-designed Data encryption and authorization systems expend considerable effort to protect passwords and other credentials from being stolen. Transferring and storing passwords in plaintext form leaves them at risk of exposure to attackers, eavesdroppers and spyware. In order to avoid such exposure, powerful encryption/authentication systems use various mechanisms to minimize the possibility that unencrypted credentials will be exposed, as well as be sure that any authentication data that does get transmitted and stored will be of minimal use to an attacker. In this we are using Encryption technique for file encryption and will send this file to other client and a server to store the data in case of data lose.

TABLE OF CONTENTS

S.No. Contents

INTRODUCTION

Key based encryption can be categorized into two types: symmetric key and asymmetric key. The algorithms of symmetric-key, known as one-key, single-key, and private key encryption are a type of cryptography that uses a public and private algorithm to implement encryption/decryption. Commonly used symmetric algorithms include Blowfish, DES, TEA, TDES, IDEA, CAST5, AES (Rijndael), RC6, Two Fish, MARS and Serpent. The asymmetric key algorithms, also referred to as public key encryption, use two keys, one called the public key and on called the private key. Common asymmetric algorithms include: RSA, PGP, SSH and SSL. In addition to the previous two types of encryption, there is a third type called hash which focuses on securing authentication. However traditional algorithms of this kind (such as SHA and MD5) have gradually become obsolete. In spite of their improvements, once an attacker is able to get a hash and successfully provide it to the authentication server, he can assume whatever security identity is associated with this hash authors implemented image and text encryption along with decryption. The text encryption uses 128-bit key size and also plaintext. Every word or space is changed into an 8-bit sequence. Thus, maximum overall 16 positions are identified by this code. AES encryption algorithm in CFB mode is used for image encryption. The PKCS5Padding method is used. Results verify the superior speed of AES. In , authors presented a comparison study of encryption standards AES, DES and RSA considering different parameters Engineering, Technology & Applied Science ResearchVol. 7, No. 4, 2017, 178117851782 www.etasr.com Harba: Secure Data Encryption Through a Combination of AES, RSA and HMAC including memory usages and computation time. A cryptographic tool has been used for testing. Depending on the text files used and the experimental result it is shown that AES and DES algorithm uses minimum encryption time compare to RSA. In addition, DES has the lowest memory usage, and the time of encryption between AES and DES algorithm has a very minor difference. When datasize increases then the asymmetric cryptographic algorithm performs slower compare to the symmetric algorithm. In, authors presented a new approach to saving files in the cloud. They used RSA and AES algorithms for protecting data and connection based upon various keys in encryption and decryption. They

also used an SHA1 algorithm to protect the hash table of data. Their model handles the total cloud system security to protect data. AES is used for signing in and RSA to encrypt data on storage. The SHA1 function is used to hash the key on the system. A key managing center is used as a third party to send out keys in all stages.

In the present paper, three methods are combined: a symmetric-key algorithm type (AES) has been used to encrypt data, standard type asymmetric cryptography (RSA) has been used to encrypt AES key and hash type (HMAC) has been used in between the two sides (that share a secret key) in order to authenticate transferred information.

SYSTEM REQUIREMENTS

Hardware Interface:

Minimum requirements will be as follows:

1.256 MB RAM required

2.Processor with speed of 500 MHz

3.Internet or LAN connection **Software interface:**

- Windows 10 / Linux
- Database
- Java development kit
- Eclipse IDE
- Notepad ++

DESIGN

3.1 flowchart

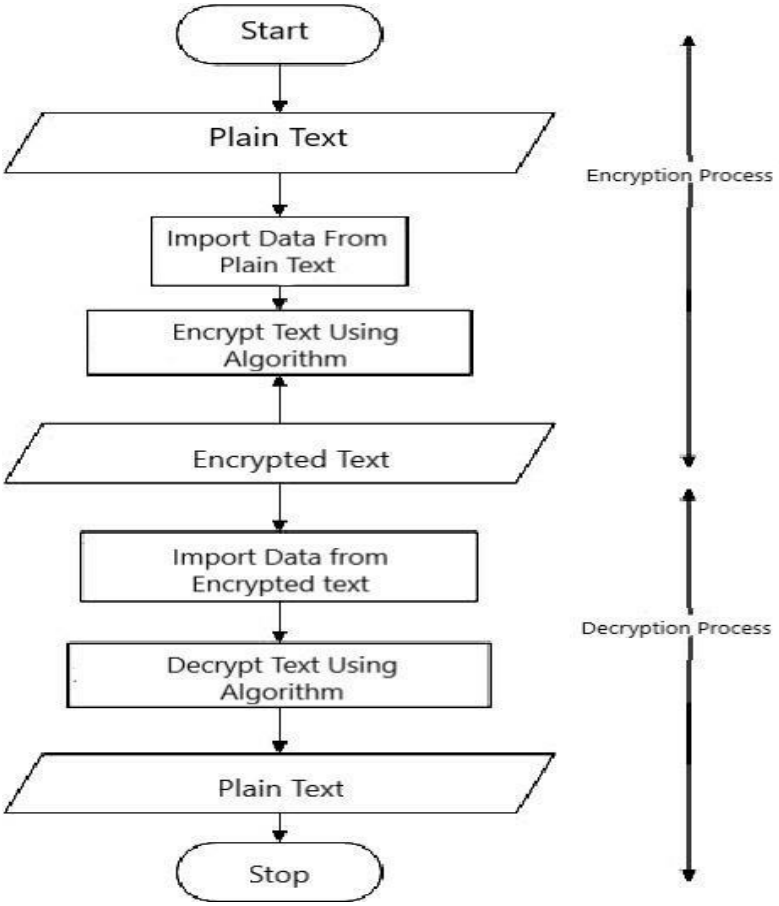


Figure 1:- Flow Chart

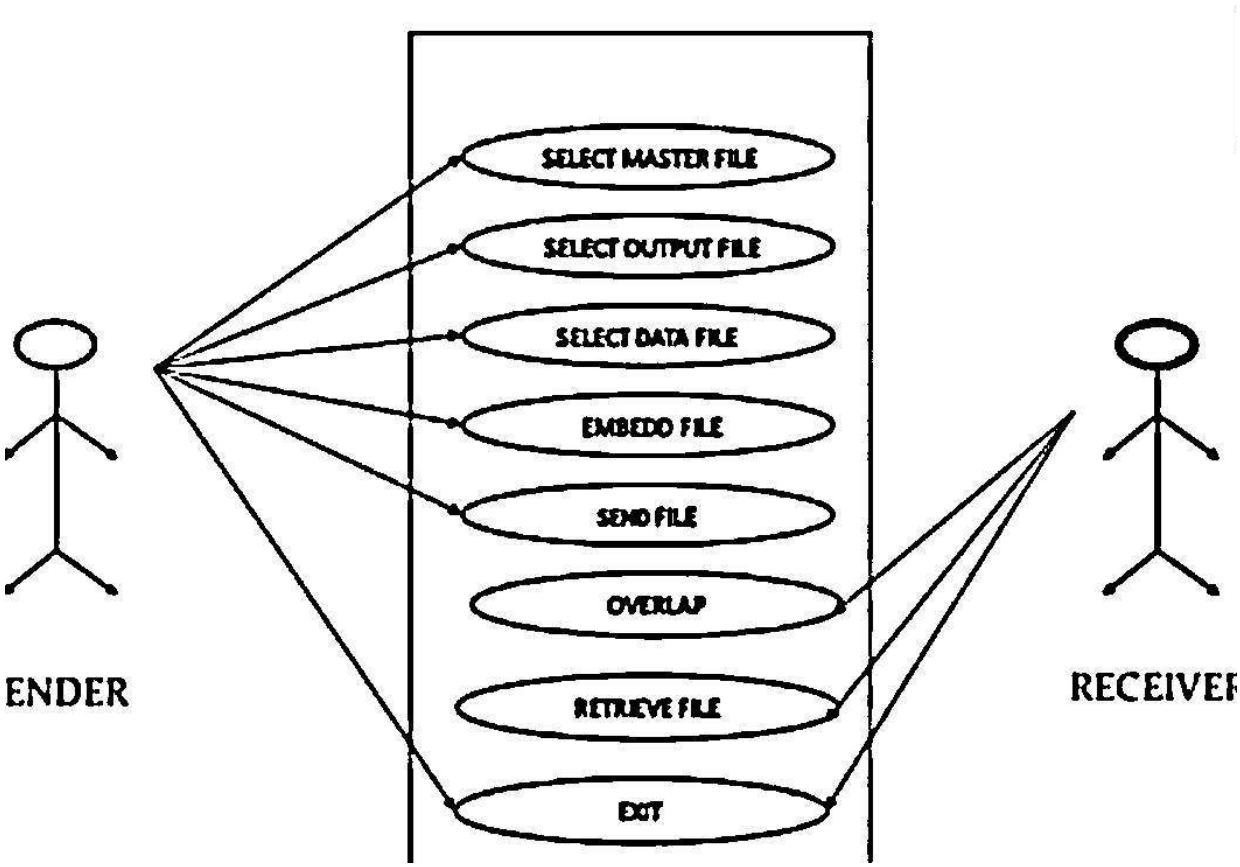


Figure 2 : UML DIAGRAM

4. Problem Statement

The client-server communication model is used in a wide variety of software applications. Where normally the server side is sufficiently protected and sealed from public access, but client applications running on devices like notebooks and desktops are considered insecure and exposed to security threats. The main weakness of client-server application is that there is no security provided to data which is transferred between clients. Any unauthorized client can hack the client

account and can change the data. This is the main objective of this project (To secure Data Transfer between Client and Server)

5. Literature Review

Socket is a standard connection protocol that supports data communication over the network between connected terminals. The standard connection supports the data transmission both by the TCP and UDP protocols between the terminals. TCP is a transport layer protocol used by applications that require guaranteed delivery of data. Basically, it is a connection-oriented protocol. To communicate over TCP one must first have to establish a connection between pair of sockets, where one socket is client and the other belongs to server. After the connection is established between them then they can communicate with each other.

A client is a system that accesses or desires for a service made accessible by a server. A server is a system (hardware or software) program running to provide the service requests of other system programs. Port is a software mechanism that allows the centralized connected Servers to listen for requests made by clients. Port is actually purposed as a gateway to listen for the requested parameters by the server terminals or other machines. It is a software address on a system that is on the network. Entire request response proceeding among this Application is carries through machine ports.

It is more convenient to use peer-to-peer application because it operates directly to the clients. The communication to the servers consumes more time if compared to the peer-to-peer. That is why peer-to-peer is more preferred to be used for Data Transfer systems as main technique and also as an alternative method to use especially for the purpose of file transferring feature.

A database is an organized collection of structured information, or data, typically stored electronically in a computer system. A database is usually controlled by a database management system (DBMS). Together, the data and the DBMS, along with the applications that are associated with them, are referred to as a database system, often shortened to just database.

Data within the most common types of databases in operation today is typically modeled in rows and columns in a series of tables to make processing and data querying efficient. The data can then

be easily accessed, managed, modified, updated, controlled, and organized. Most databases use structured query language (SQL) for writing and querying data.

6.Objectives

The aim of this project is to develop a reliable and secure network programming (Data Transfer) using socket programming and to implement peer to peer concept. The data sent from the sender side is encrypted and is then decrypted at the receiver side to read the message and ensure the safety and confidentiality of the message and ensure the safety, confidentiality and Integrity of the File and will store the data on server in case of data lose.

7.1.1 Limitations of AES

Tackling of the Error propagation of Advanced Encryption Standard (AES) is a great challenge. In literature, several studies have been made on this issue and several techniques are suggested to tackle the effect. Error propagation effect in case of selective AES and its comparison with normal AES has also been studied. Four algorithms, viz. SBM 1.1, SBM 1.2, SBM 1.3 and SBM 1.4, have been proposed for preventing error propagation effect of AES.

Apart from the error propagation effect, AES may suffer from different attacks made over the channel. Larger key lengths translate into an exponential increase in the complexity of an exhaustive search. Side-channel attacks, however, use a divide-and-conquer approach [104, 105] and hence it is generally assumed that increasing the key length cannot be used as mitigation. Yet, the internal round structure of AES-256 and its key-scheduling seem to hinder a direct extension of the existing attacks thus challenge the proposition above. Indeed two consecutives round keys are required to infer the secret key and the MixColumns operation, not present in the last round, apparently increases the key search complexity from to 28 to 232. Additionally, it is unclear what the impact of the different round structures is on the number of required measurements.

7.1.2 Limitations of RSA

RSA, as we saw is a really amazing public key cipher that uses only basic number theory in its description. However, whenever a new cipher appears there will be many people that test its security and whenever possible will try to break it. So far RSA has not been broken but certain bad things can happen with it if you are not careful. Here are a few things that can go wrong. Using small primes. Using primes that are very close. Message is an observable eth power. Two people using the same N, receiving the same message. Sending the same message to eor more people with the same e.

7.2 Future Enhancements

7.2.1 AES future scope

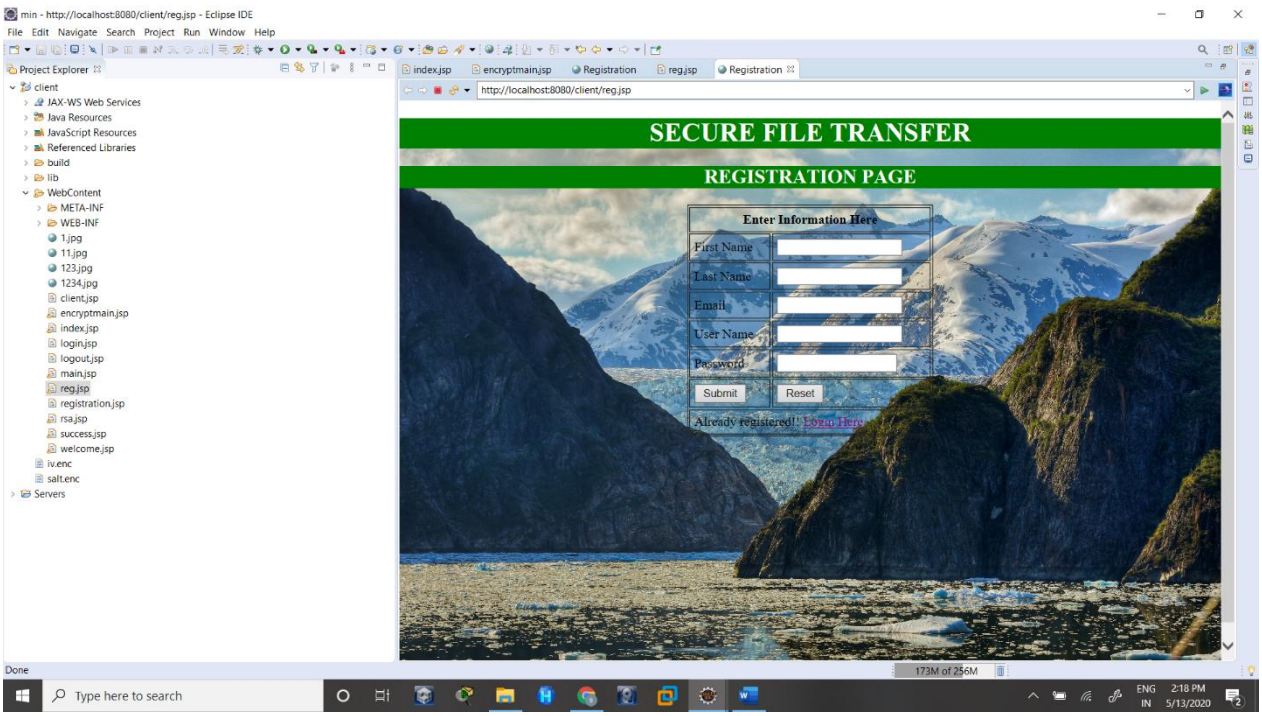
In AES algorithm, the number of rounds involved in the encryption and decryption depends on the length of the key and the number of block columns.
So, the number of rounds may be increased to improve the strength of the AES. The strength of the AES algorithm may enhanced by increasing the key length from 128 bits to 512 bits and thereby the number of rounds is increased in order to provide a stronger encryption method for secure communication.Code optimization may be done in order to improve the speed of encryption and decryption using the 512 bit AES. Enhance The speed of Encryption and Decryption.

7.2.2 RSA future scope

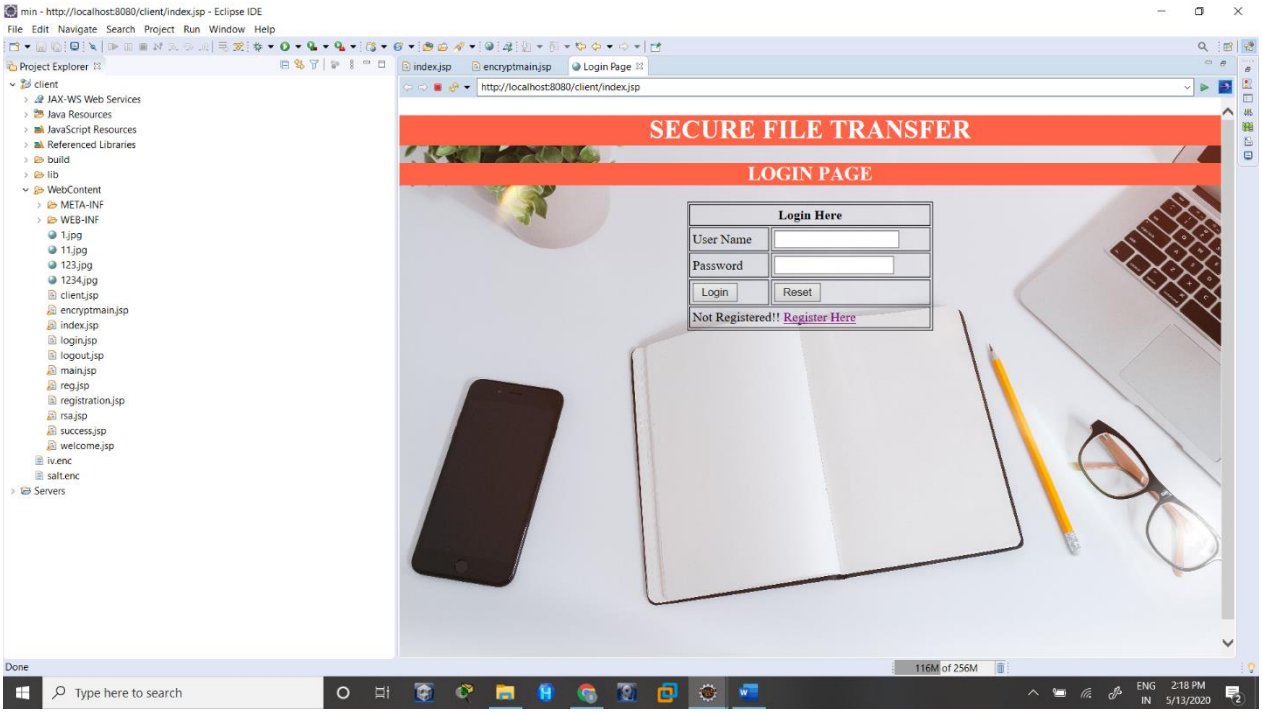
Many key problems inherent to the distributed nature ofmulticasting and to the resource constraints in sensor equipments and inwireless devices, in general, make securing group communications over thistype of networks more difficult. Among other problems one can cite:
Efficient security mechanisms for resource constrained devices
Security context and mobility impact
Absence of third trust party
Masking heterogeneity

8.ScreenShots

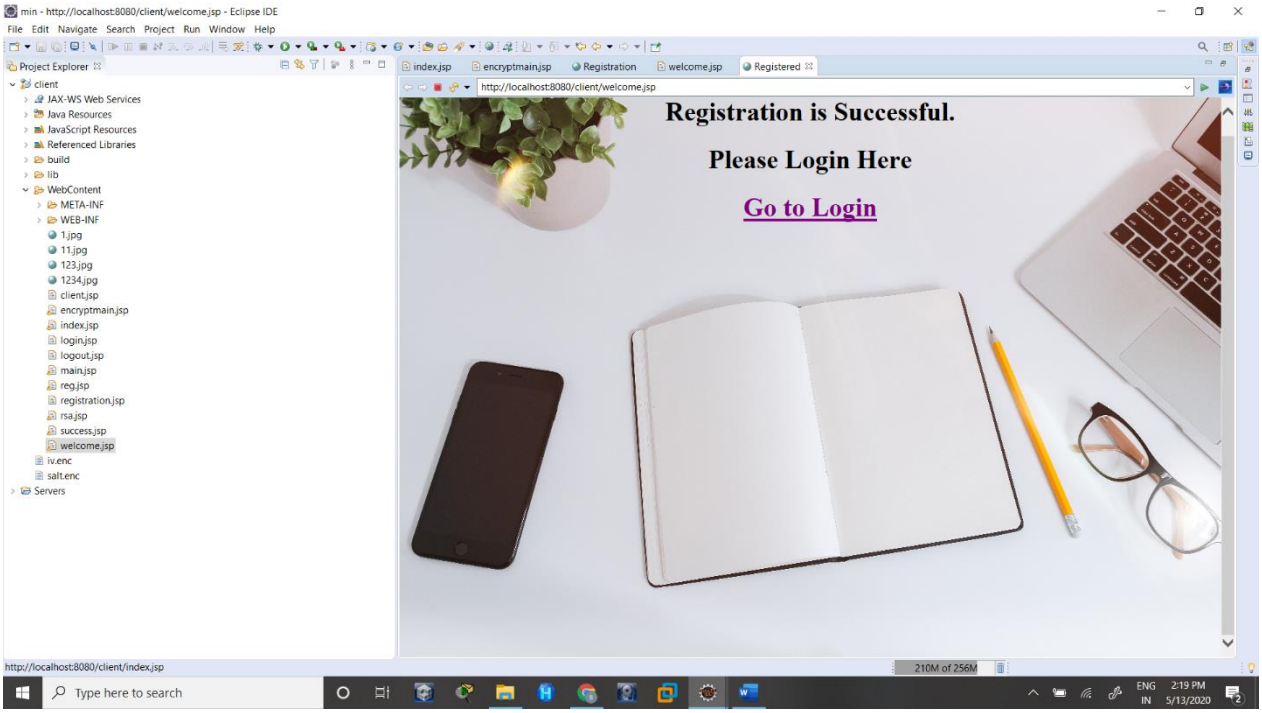
Registration page:



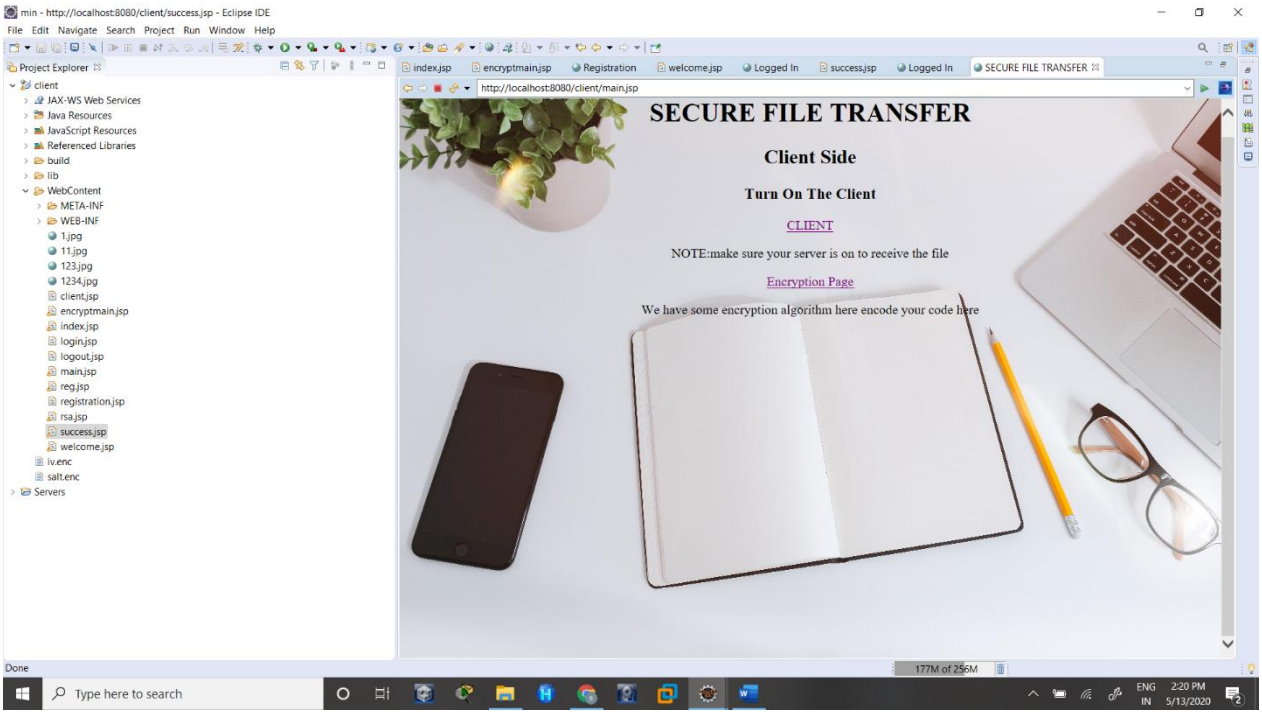
Login Page:



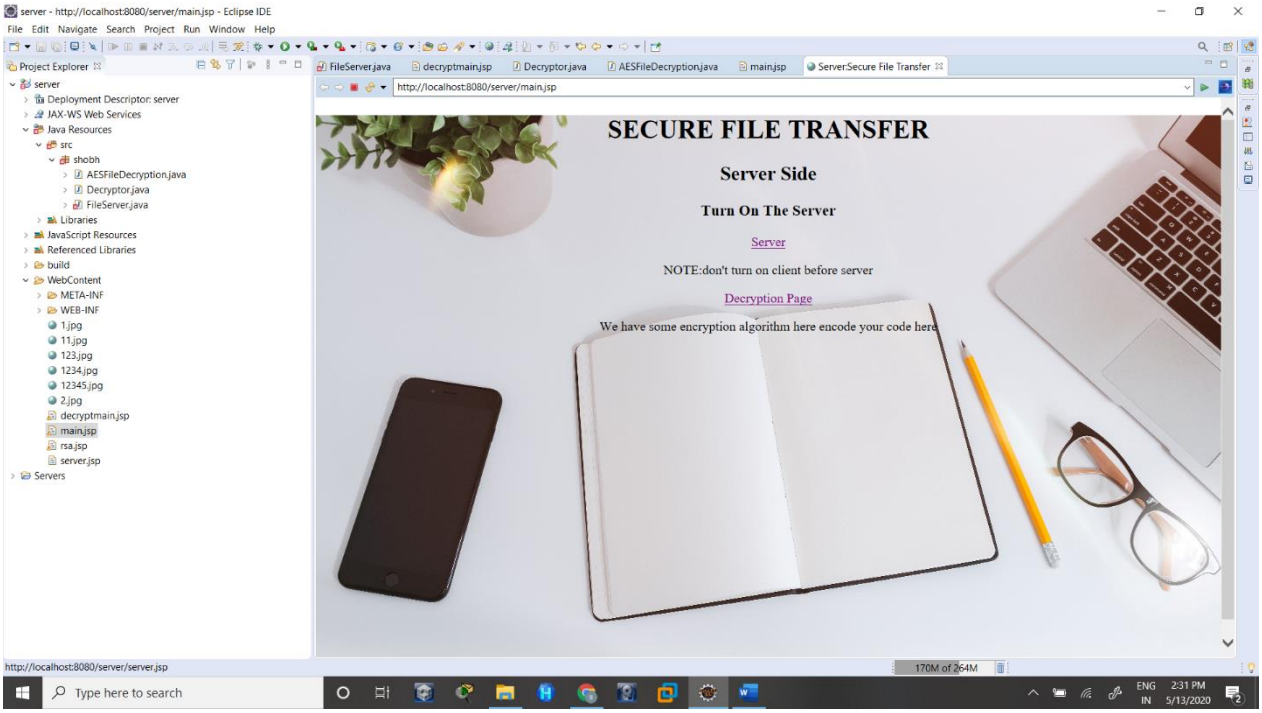
After Registration Page:



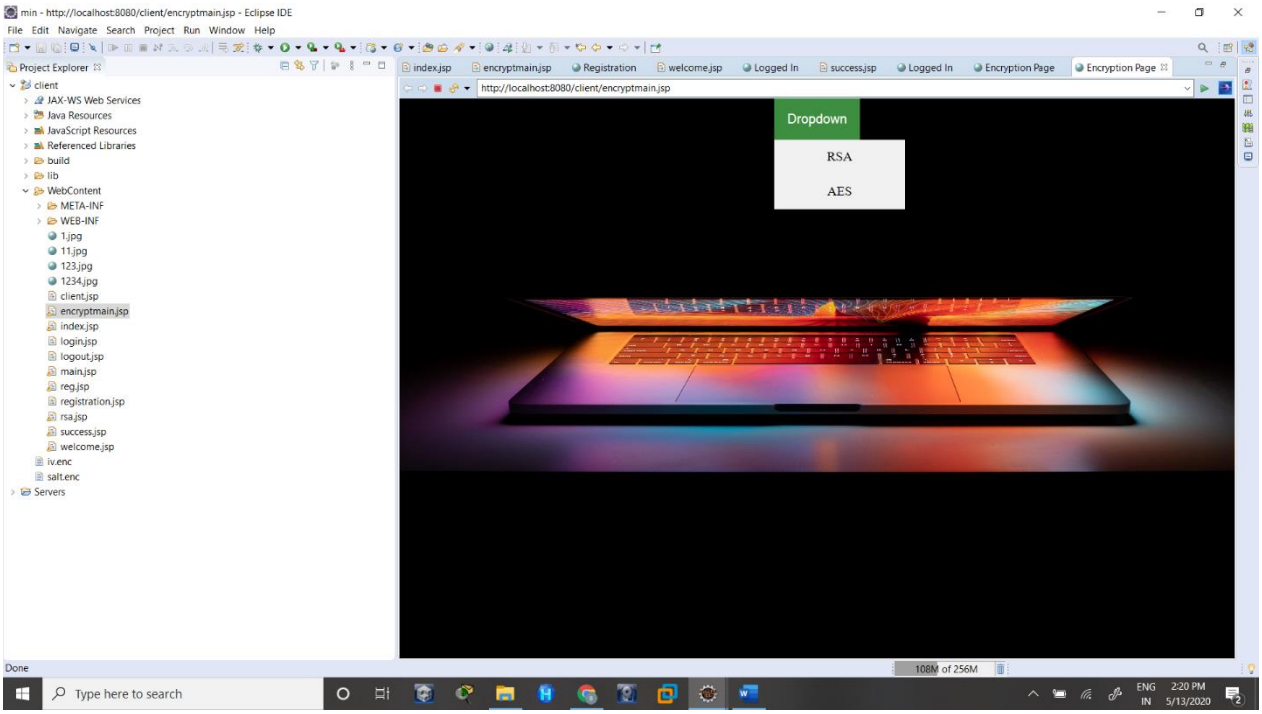
Main page: Client side



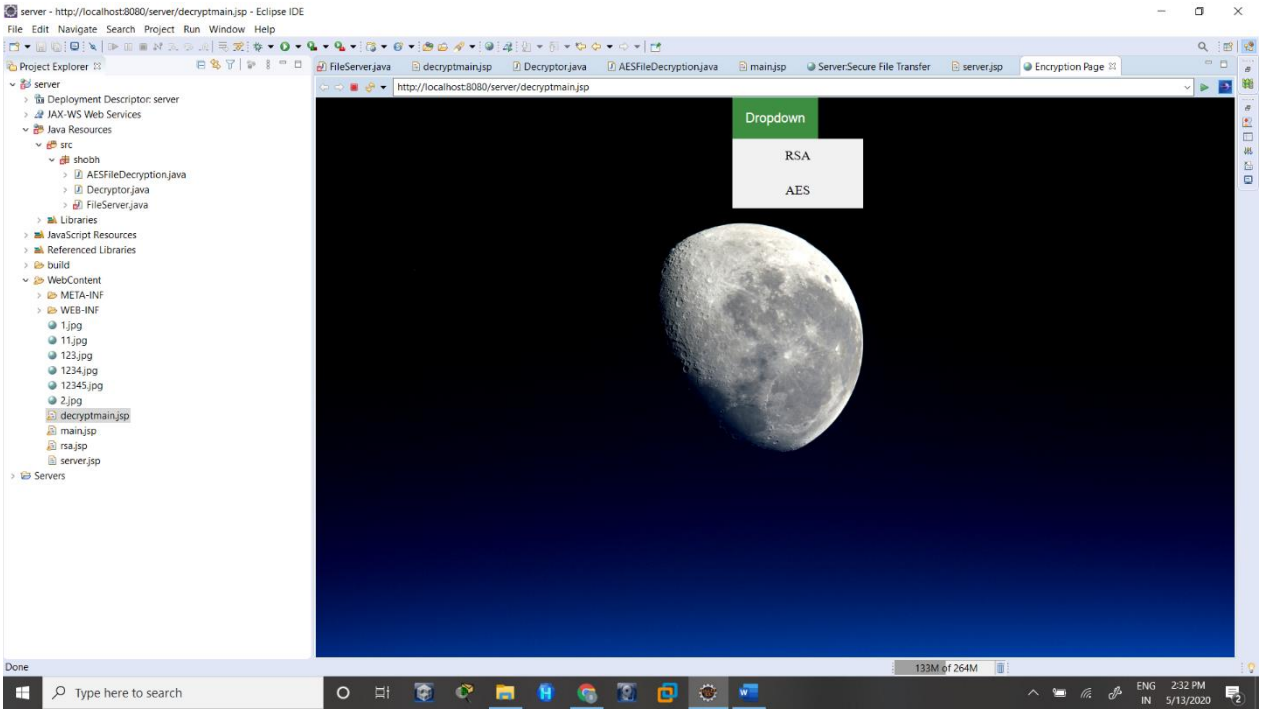
Main Page: Server side



Encryption page:

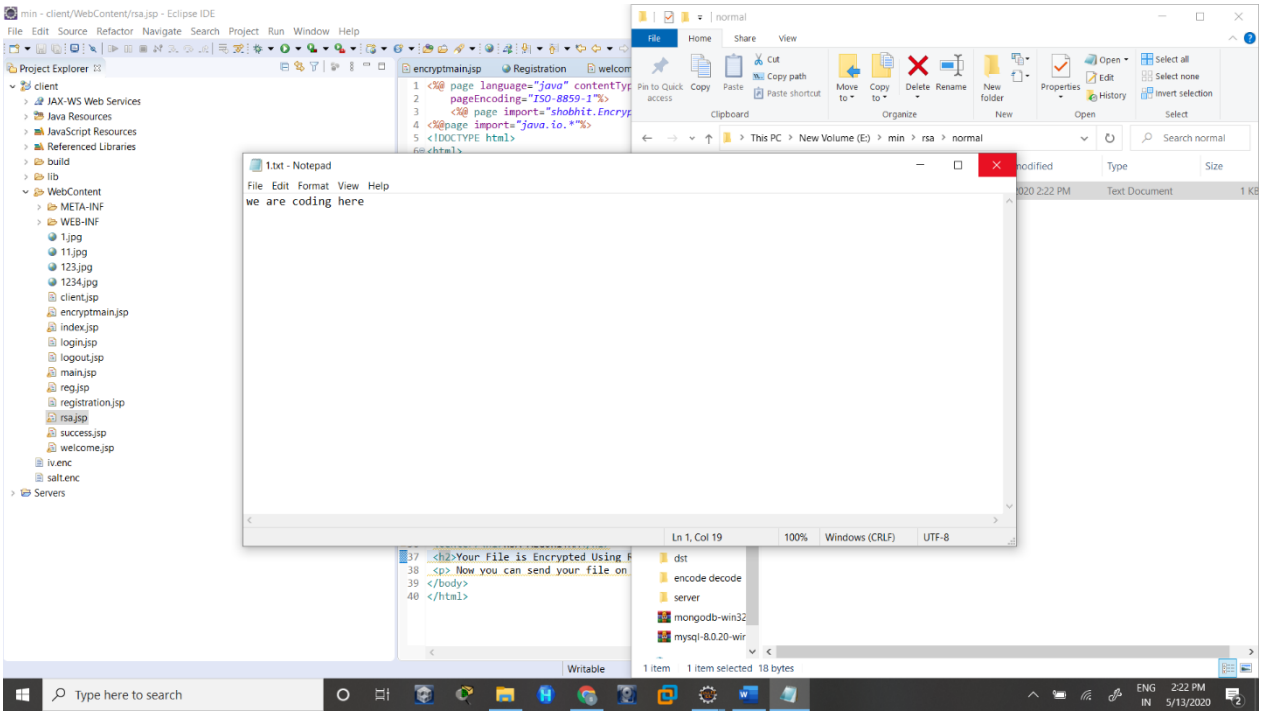


Decryption page:

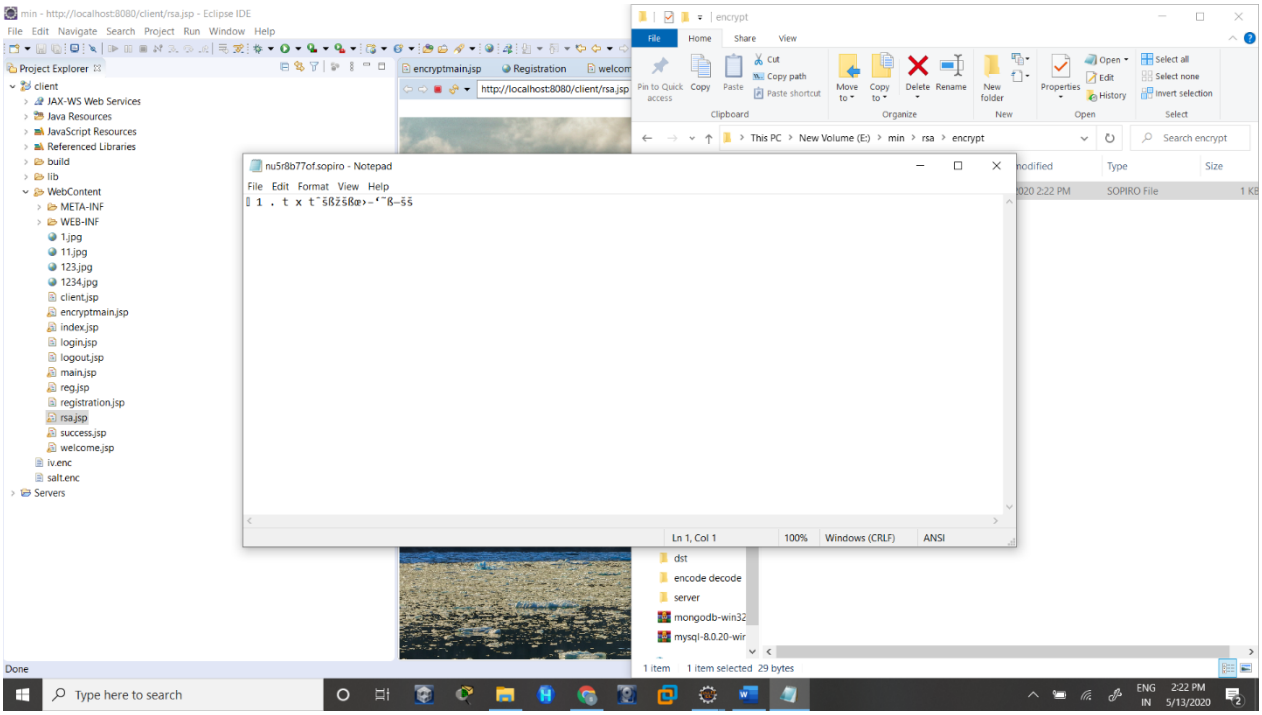


Rsa Encryption:

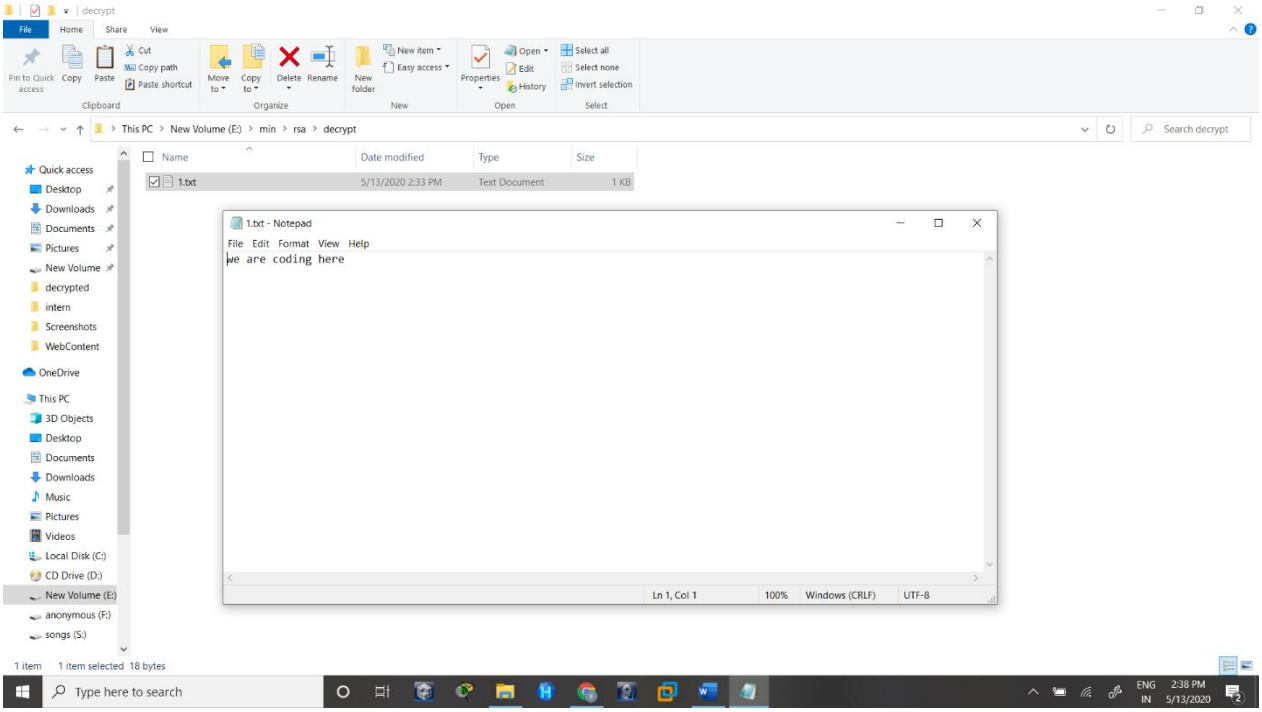
Before encryption:



After encryption:

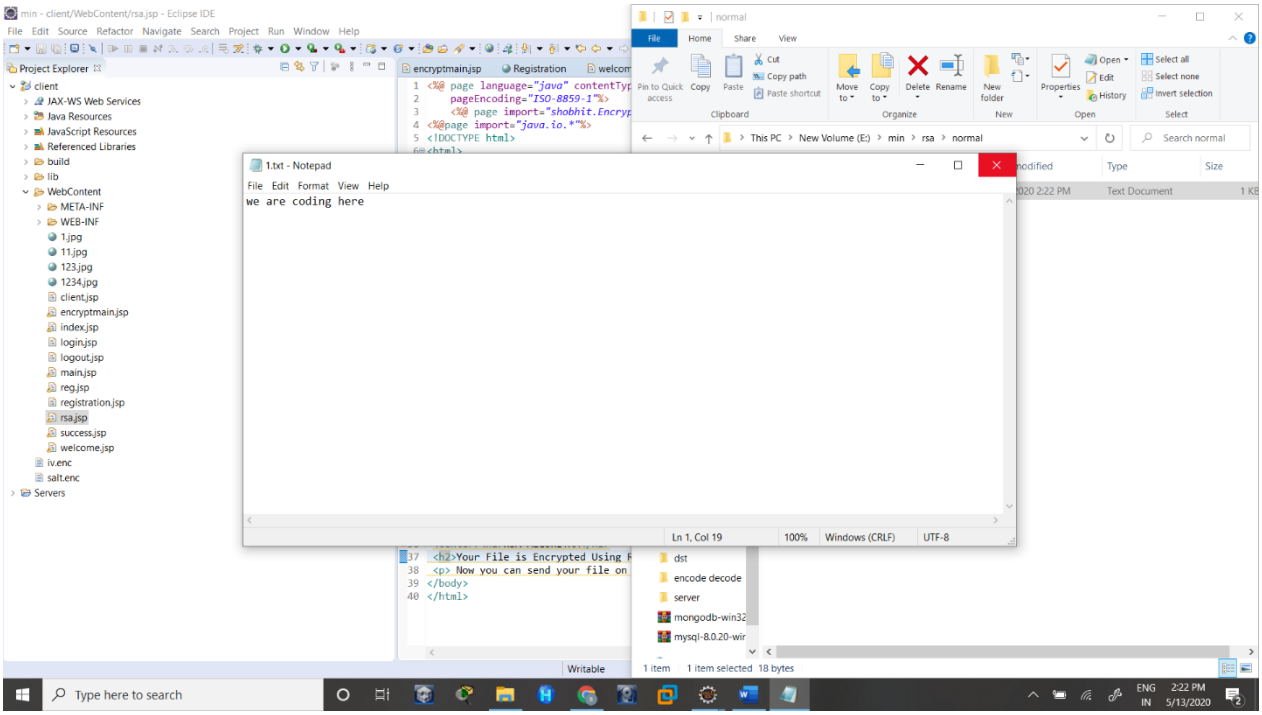


Decryption:

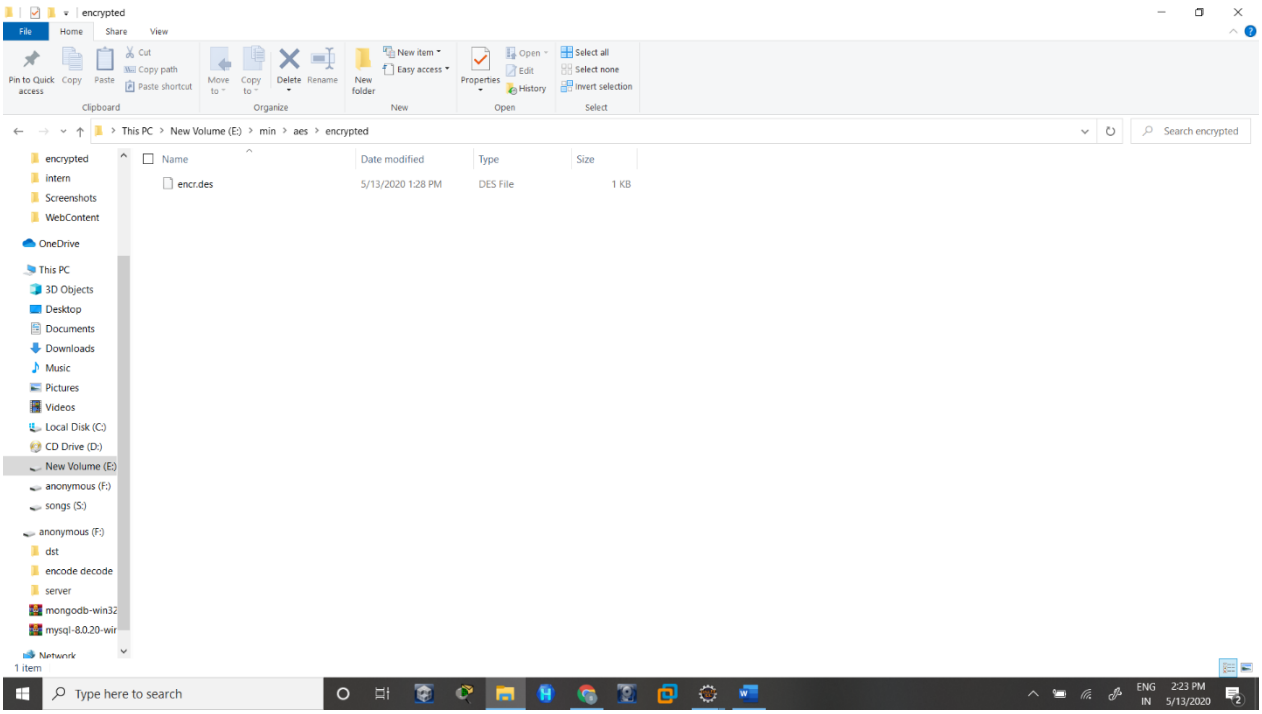


AES Encryption and Decryption

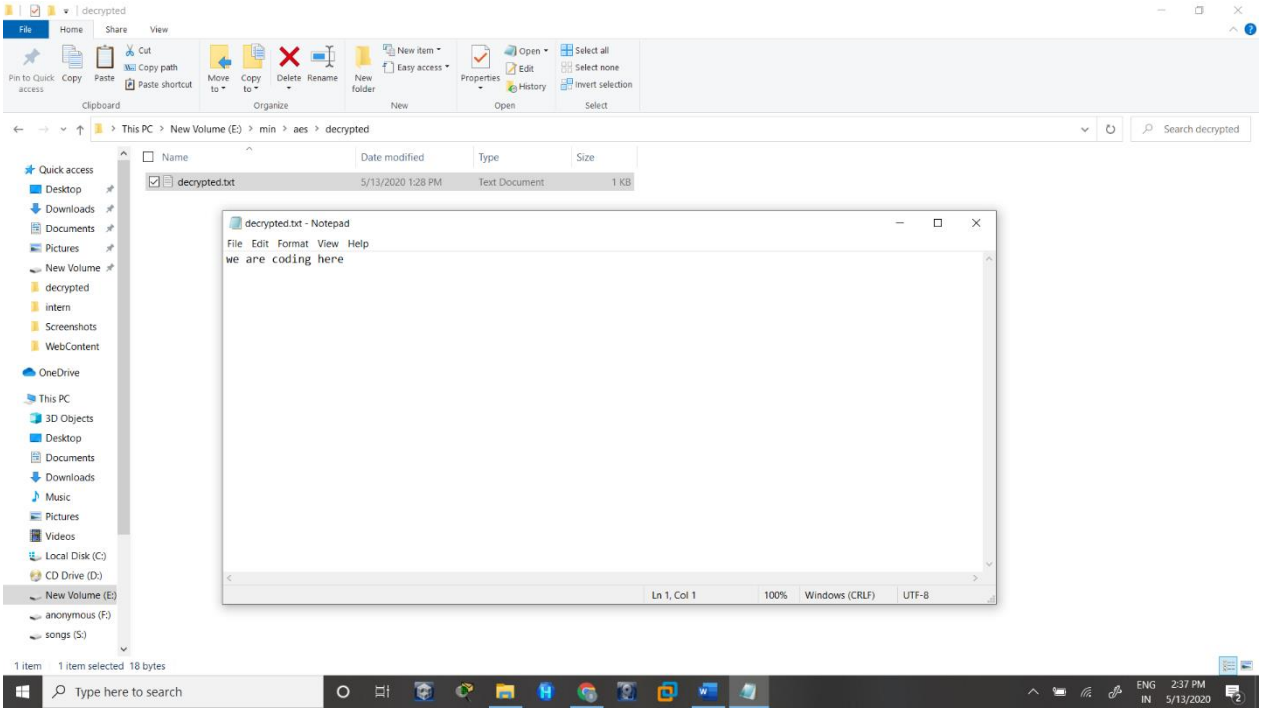
Before:



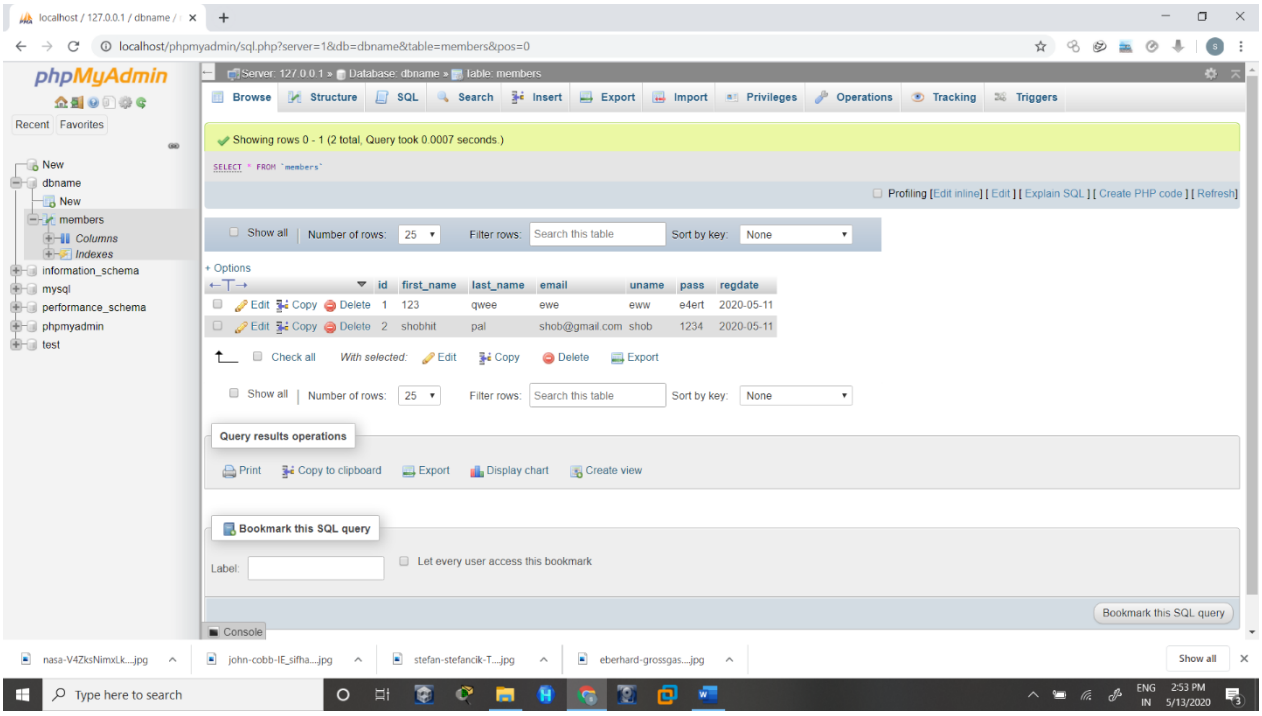
After:



Decryption:



DataBase:



9. Conclusion

Applied AES (256bit) algorithm Successfully.

Applied RSA algorithm successfully.

References:

1. Limi Kalita," Socket Programming", *(IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 5 (3) , 2014, 4802-4807.*
2. Pranab Bandhu Nath & Md.Mofiz Uddin," TCP-IP Model in Data Communication and Networking", *American Journal of Engineering Research (AJER) eISSN: 2320-0847 p-ISSN : 2320-0936.*
3. Rushabh Balpande, Chetan Dusane, Khushboo Kashyap & Nandkumar Patil,"Client Server Based Secure Data Transfer Application Using Peer-ToPeer Network Architecture" *International Journal of Emerging Trend in Engineering and Basic Sciences (IJEEBS) ISSN (Online) 2349-6967*
4. By Dr. Prerna Mahajan & Abhishek Sachdeva," A Study of Encryption Algorithms AES, DES and RSA for Security", *Global Journal of Computer Science and Technology Network, Web & Security Volume 13 Issue 15 Version 1.0 Year 2013 Online ISSN: 0975-4172 & Print ISSN: 0975-4350*

Annexure

CLIENT SIDE

Index.jsp

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
    <style>
body, html {
  height: 100%;
  margin: 0;
}

.bg {
  /* The image used */
  background-image: url("11.jpg");

  /* Full height */
  height: 100%;

  /* Center and scale the image nicely */
  background-position: center;
  background-repeat: no-repeat;
  background-size: cover;
}
</style>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Login Page</title>
    </head>
    <body><div class="bg">
    <center><h1 style="background-color:tomato;color:white">SECURE FILE TRANSFER</h1>
    <h2 style="background-color:tomato;color:white">LOGIN PAGE</h2>
    </center>
        <form method="post" action="login.jsp">
            <center>
            <table border="1" width="30%" cellpadding="3">
            <thead>
                <tr>
                    <th colspan="2">Login Here</th>
                </tr>
            </thead>
            <tbody>
                <tr>
                    <td>User Name</td>
                    <td><input type="text" name="uname" value="" /></td>
                </tr>
                <tr>
                    <td>Password</td>
                    <td><input type="password" name="pass" value="" /></td>
                </tr>
                <tr>
```

```
                <td><input type="submit" value="Login" /></td>
                <td><input type="reset" value="Reset" /></td>
            </tr>
            <tr>
                <td colspan="2">Not Registered!! <a href="reg.jsp">Register Here</a></td>
            </tr>
        </tbody>
    </table>
    </center>
    </form></div>
  </body>
</html>
```

login.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
  pageEncoding="ISO-8859-1"%>
  <%@ page import ="java.sql.*" %>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<%
  String userid = request.getParameter("uname");
  String pwd = request.getParameter("pass");
  Class.forName("com.mysql.jdbc.Driver");
  Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/dbname",
      "root","");
  Statement st = con.createStatement();
  ResultSet rs;
  rs = st.executeQuery("select * from members where uname='" + userid + "' and pass='" + pwd + "'");
  if (rs.next()) {
    session.setAttribute("userid", userid);
    //out.println("welcome " + userid);
    //out.println("<a href='logout.jsp'>Log out</a>");
    response.sendRedirect("success.jsp");
  } else {
    out.println("Invalid password <a href='index.jsp'>try again</a>");
  }
%>
</body>
</html>
```

reg.jsp

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>  <style>
body, html {
 height: 100%;
 margin: 0;
```

```
}

.bg {
  /* The image used */
  background-image: url("11.jpg");

  /* Full height */
  height: 100%;

  /* Center and scale the image nicely */
  background-position: center;
  background-repeat: no-repeat;
  background-size: cover;
}
</style>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Registration</title>
    </head>
    <body><div class="bg">
    <center><h1 style="background-color:green;color:white">SECURE FILE TRANSFER</h1>
    <h2 style="background-color:green;color:white">REGISTRATION PAGE</h2></center>
        <form method="post" action="registration.jsp">
            <center>
        <table border="1" width="30%" cellpadding="5">
            <thead>
                <tr>
                    <th colspan="2">Enter Information Here</th>
                </tr>
            </thead>
            <tbody>
                <tr>
                    <td>First Name</td>
                    <td><input type="text" name="fname" value="" /></td>
                </tr>
                <tr>
                    <td>Last Name</td>
                    <td><input type="text" name="lname" value="" /></td>
                </tr>
                <tr>
                    <td>Email</td>
                    <td><input type="text" name="email" value="" /></td>
                </tr>
                <tr>
                    <td>User Name</td>
                    <td><input type="text" name="uname" value="" /></td>
                </tr>
                <tr>
                    <td>Password</td>
                    <td><input type="password" name="pass" value="" /></td>
                </tr>
                <tr>
                    <td><input type="submit" value="Submit" /></td>
                    <td><input type="reset" value="Reset" /></td>
                </tr>
                <tr>
                    <td colspan="2">Already registered!! <a href="index.jsp">Login Here</a></td>
```

```
            </tr>
          </tbody>
        </table>
        </center>
      </form></div>
   </body>
</html>
```

Registration.jsp

```
<%@ page import ="java.sql.*" %>
<%
   String user = request.getParameter("uname");
   String pwd = request.getParameter("pass");
   String fname = request.getParameter("fname");
   String lname = request.getParameter("lname");
   String email = request.getParameter("email");
   String username="root";
   String connUrl="jdbc:mysql://localhost:3306/dbname";
   Class.forName("com.mysql.jdbc.Driver");
   Connection con = DriverManager.getConnection(connUrl,username,"");
   Statement st = con.createStatement();
   //ResultSet rs;
   int i = st.executeUpdate("insert into members(first_name, last_name, email, uname, pass, regdate) values ('" +
fname + "','" + lname + "','" + email + "','" + user + "','" + pwd + "', CURDATE())");
   if (i > 0) {
      //session.setAttribute("userid", user);
      response.sendRedirect("welcome.jsp");
     // out.print("Registration Successfull!"+"<a href='index.jsp'>Go to Login</a>");
   } else {
      response.sendRedirect("index.jsp");
   }
%>
```

Welcome.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head><style>
body, html {
  height: 100%;
  margin: 0;
}

.bg {
  /* The image used */
  background-image: url("11.jpg");

  /* Full height */
  height: 100%;

  /* Center and scale the image nicely */
  background-position: center;
```

```
  background-repeat: no-repeat;
  background-size: cover;
}
</style>
<meta charset="ISO-8859-1">
<title>Registered</title>
</head>
<body>
<div class="bg">
<center>
<h1>Registration is Successful.</h1>
<h1>Please Login Here </h1>
<h1><a href='index.jsp'>Go to Login</a></h1>
</center>
</div>
</body>
</html>
```

Success.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head><style>
body, html {
  height: 100%;
  margin: 0;
}

.bg {
  /* The image used */
  background-image: url("11.jpg");

  /* Full height */
  height: 100%;

  /* Center and scale the image nicely */
  background-position: center;
  background-repeat: no-repeat;
  background-size: cover;
}
</style>
<meta charset="ISO-8859-1">
<title>Logged In</title>
</head>
<body><div class="bg"><center>
<%
    if ((session.getAttribute("userid") == null) || (session.getAttribute("userid") == "")) {
%>
<p style="background-color:red;color:white">You are not logged in</p><br/>
<a href="index.jsp">Please Login</a>
<%} else {
```

```
%>
<p style="background-color:blue;color:white">Welcome <%=session.getAttribute("userid")%></p>
<p style="background-color:blue;color:white">click the link to go</p>
<pre><a href="main.jsp">FILE TRANSFER</a></pre>
<pre><a href='logout.jsp'>Log out</a></pre>
<%
    }
%></center>
</div>
</body>
</html>
```

main.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<style>
body, html {
  height: 100%;
  margin: 0;
}

.bg {
  /* The image used */
  background-image: url("11.jpg");

  /* Full height */
  height: 100%;

  /* Center and scale the image nicely */
  background-position: center;
  background-repeat: no-repeat;
  background-size: cover;
}
</style>

<meta charset="ISO-8859-1">
<title>SECURE FILE TRANSFER</title>
</head>
<body>
<div class="bg">

<center><h1>SECURE FILE TRANSFER</h1></center>
<center><h2>Client Side</h2></center>
<center><h3>Turn On The Client</h3></center>
<center><a href="client.jsp">CLIENT</a></center>
<center><p>NOTE:make sure your server is on to receive the file</center>
<center><a href="encryptmain.jsp">Encryption Page</a>
    <p>We have some encryption algorithm here encode your code here</p>

</center>
```

```
</form>

</div>

</body>
</html>
```

Client.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@ page import="shobhit.FileClient" %>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<% FileClient fc = new FileClient("localhost", 1988, "E:\\minor2_program-codes_grp16\\");



%>

</body>
</html>
```

FileClient.java

```
package shobhit;


import java.io.DataOutputStream;

import java.io.FileInputStream;

import java.io.IOException;

import java.net.Socket;


public class FileClient {


	private Socket s;


	public FileClient(String host, int port, String file) {

		try {

			s = new Socket(host, port);
```

```
                    sendFile(file);
                } catch (Exception e) {
                    e.printStackTrace();
                }
        }

        public void sendFile(String file) throws IOException {
                DataOutputStream dos = new DataOutputStream(s.getOutputStream());
                FileInputStream fis = new FileInputStream(file);
                byte[] buffer = new byte[4096];

                while (fis.read(buffer) > 0) {
                    dos.write(buffer);

                }

                fis.close();
                dos.close();
        }
}
```

Encryptmain.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<style>
body, html {
  height: 100%;
  margin: 0;
}

.bg {
  /* The image used */
  background-image: url("11.jpg");

  /* Full height */
  height: 100%;
```

```
  /* Center and scale the image nicely */
  background-position: center;
  background-repeat: no-repeat;
  background-size: cover;
}

.dropbtn {
  background-color: #4CAF50;
  color: white;
  padding: 16px;
  font-size: 16px;
  border: none;
}

.dropdown {
  position: relative;
  display: inline-block;
}

.dropdown-content {
  display: none;
  position: absolute;
  background-color: #f1f1f1;
  min-width: 160px;
  box-shadow: 0px 8px 16px 0px rgba(0,0,0,0.2);
  z-index: 1;
}

.dropdown-content a {
  color: black;
  padding: 12px 16px;
  text-decoration: none;
  display: block;
}

.dropdown-content a:hover {background-color: #ddd;}

.dropdown:hover .dropdown-content {display: block;}

.dropdown:hover .dropbtn {background-color: #3e8e41;}
</style>

<meta charset="ISO-8859-1">
<title>Encryption Page</title>
</head>
<body>
<div class="bg"><center>
<div class="dropdown">
  <button class="dropbtn">Dropdown</button>
  <div class="dropdown-content">
    <a href="rsa.jsp">RSA</a>
    <a href="#">AES</a>
  </div>
</div></center>

</div>
```

```
</body>
</html>
```

rsa.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
    <%@ page import="shobhit.Encryptor" %>
<%@page import="java.io.*"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<%Encryptor en = Encryptor.getEncrypter(true);
 String strpath="E:/min/rsa/normal";
 String strpath1="E:/min/rsa/encrypt";
 File src = new File(strpath);
 File dst=new File(strpath1);
 en.encrypt(src, dst);  %>
 <center><h1>RSA ALGORITHM</h1>
 <p>Your File is Encrypted Using RSA Algorithm and Encrypted File is Stored at Destination</p>
 <p> Now you can send your file on the network</p></center>
</body>
</html>
```

Encryptor.java

```
package shobhit;



import java.io.File;

import java.io.FileInputStream;

import java.io.FileOutputStream;

import java.io.IOException;

import java.io.InputStream;

import java.io.OutputStream;

import java.util.Random;



public class Encryptor

{

        private static Encryptor encrypter = new Encryptor();
```

```
private static boolean deleteOriginal;

public static Encryptor getEncrypter(boolean originalFileDeleted)
{
	deleteOriginal = originalFileDeleted;

	return encrypter;
}

public void encrypt(File src, File dst)
{
	if (!dst.exists())
		dst.mkdir();
	if (!dst.isDirectory())
		return;

	try
	{
		if (!src.isDirectory())
		{
			copyEncrypted(src, dst);
		} else
		{
			File[] files = src.listFiles();

			System.out.println("Encrypting...");

			for (File f : files)
			{
				copyEncrypted(f, dst);
				if(deleteOriginal) f.delete();
```

```
                        }

                        System.out.println(files.length + " files are encrypted");
                }
        } catch (IOException e)
        {
                e.printStackTrace();
        }
}

public void copyEncrypted(File source, File dest) throws IOException
{
        InputStream is = null;
        OutputStream os = null;

        dest = new File(dest.getPath().concat("/").concat(getRandomName(10, "sopiro")));

        try
        {
                is = new FileInputStream(source);
                os = new FileOutputStream(dest);

                os.write(new byte[] { (byte) source.getName().length() });
                os.write(stringToByte(source.getName()));

                byte[] buffer = new byte[1024];

                int length;

                while ((length = is.read(buffer)) > 0)
                {
                        encryptBytes(buffer);
```

```
                os.write(buffer, 0, length);
            }

        } finally
        {
            is.close();
            os.close();
        }
}

private void encryptBytes(byte[] data) // Encryption Algorithm is written into here
{
        for (int i = 0; i < data.length; i++)
        {
            data[i] = (byte) ~data[i];
        }
}

public byte[] stringToByte(String data)
{
        char[] ca = data.toCharArray();
        byte[] res = new byte[ca.length * 2]; // Character.BYTES = 2;

        for (int i = 0; i < res.length; i++)
        {
            res[i] = (byte) ((ca[i / 2] >> (8 - (i % 2) * 8)) & 0xff);
        }

        return res;
}

public String getRandomName(int length, String extend)
```

```
{
        Random r = new Random();
        StringBuilder res = new StringBuilder();

        for (int i = 0; i < length; i++)
        {

                char c = 'a';
                int width = 'z' - 'a';

                if (r.nextInt(3) == 0)
                {
                        c = 'A';
                        width = 'Z' - 'A';
                }
                if (r.nextInt(3) == 1)
                {
                        c = '0';
                        width = '9' - '0';
                }

                res.append((char) (c + r.nextInt(width)));
        }

        res.append(".").append(extend);

        return res.toString();
}

public void copy(File source, File dest) throws IOException
{
        InputStream is = null;
```

```
                OutputStream os = null;

                try
                {
                        dest = new File(dest.getPath().concat("/").concat(source.getName()));

                        is = new FileInputStream(source);
                        os = new FileOutputStream(dest);

                        byte[] buffer = new byte[1024];

                        int length;
                        int tl = 0;

                        while ((length = is.read(buffer)) > 0)
                        {
                                tl += length;
                                os.write(buffer, 0, length);
                        }

                        System.out.println(tl + " bytes");
                } finally
                {
                        is.close();
                        os.close();
                }
        }
}
```

AESFileEncryption.java

```
package shobhit;

import java.io.FileInputStream;
import java.io.FileOutputStream;
```

```
import java.security.AlgorithmParameters;
import java.security.SecureRandom;
import java.security.spec.KeySpec;

import javax.crypto.Cipher;
import javax.crypto.SecretKey;
import javax.crypto.SecretKeyFactory;
import javax.crypto.spec.IvParameterSpec;
import javax.crypto.spec.PBEKeySpec;
import javax.crypto.spec.SecretKeySpec;

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

		// file to be encrypted
		FileInputStream inFile = new FileInputStream("E:\\min\\aes\\plain\\1.txt");

		// encrypted file
		FileOutputStream outFile = new FileOutputStream("E:\\min\\aes\\encrypted\\encr.des");

		// password to encrypt the file
		String password = "javapapers";

		// password, iv and salt should be transferred to the other end
		// in a secure manner

		// salt is used for encoding
		// writing it to a file
		// salt should be transferred to the recipient securely
		// for decryption
		byte[] salt = new byte[8];
		SecureRandom secureRandom = new SecureRandom();
		secureRandom.nextBytes(salt);
		FileOutputStream saltOutFile = new FileOutputStream("E:\\min\\salt.enc");
		saltOutFile.write(salt);
		saltOutFile.close();

		SecretKeyFactory factory = SecretKeyFactory
				.getInstance("PBKDF2WithHmacSHA1");
		KeySpec keySpec = new PBEKeySpec(password.toCharArray(), salt, 65536,
				256);
		SecretKey secretKey = factory.generateSecret(keySpec);
		SecretKey secret = new SecretKeySpec(secretKey.getEncoded(), "AES");

		//
		Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5Padding");
		cipher.init(Cipher.ENCRYPT_MODE, secret);
		AlgorithmParameters params = cipher.getParameters();

		// iv adds randomness to the text and just makes the mechanism more
		// secure
		// used while initializing the cipher
		// file to store the iv
		FileOutputStream ivOutFile = new FileOutputStream("E:\\min\\iv.enc");
		byte[] iv = params.getParameterSpec(IvParameterSpec.class).getIV();
		ivOutFile.write(iv);
```

```
                ivOutFile.close();

                //file encryption
                byte[] input = new byte[64];
                int bytesRead;

                while ((bytesRead = inFile.read(input)) != -1) {
                        byte[] output = cipher.update(input, 0, bytesRead);
                        if (output != null)
                                outFile.write(output);
                }

                byte[] output = cipher.doFinal();
                if (output != null)
                        outFile.write(output);

                inFile.close();
                outFile.flush();
                outFile.close();

                System.out.println("File Encrypted.");

        }

}
```

SERVER SIDE

Main.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>

<style>
body, html {
  height: 100%;
  margin: 0;
}

.bg {
  /* The image used */
  background-image: url("12.jpg");

  /* Full height */
  height: 100%;

  /* Center and scale the image nicely */
  background-position: center;
  background-repeat: no-repeat;
  background-size: cover;
}
```

```
</style>

<meta charset="ISO-8859-1">
<title>Server:Secure File Transfer</title>
</head>
<body>
<div class="bg">
<center><h1>SECURE FILE TRANSFER</h1></center>
<center><h2>Server Side</h2></center>
<center><h3>Turn On The Server</h3></center>
<center><a href="server.jsp">Server</a></center>
<center><p>NOTE:don't turn on client before server</center>
<center><a href="decryptmain.jsp">Decryption Page</a>
        <p>We have some encryption algorithm here encode your code here</p></center>

</div>
</body>
</html>
```

rsa.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
    <%@ page import="shobh.Decryptor" %>
<%@page import="java.io.*"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<%Decryptor de = Decryptor.getDecrypter(true);
 String strpath="E:/min/rsa/encrypt";
 String strpath1="E:/min/rsa/decrypt";
 File src = new File(strpath);
 File dst=new File(strpath1);
 de.decrypt(src, dst);  %>
 <center><h1>RSA ALGORITHM</h1>
 <p>Your File is Decrypted Using RSA Algorithm and Decrypted File is Stored at
Destination(E:\min\rsa\decrypt)</p>
 <p> Now you can send your file on the network</p></center>
</body>
</html>
```

server.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@ page import="shobh.FileServer" %>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
```

```
</head>
<body>
<% FileServer fs = new FileServer(1988);  %>
<% fs.start(); %>
</body>
</html>
```

decryptmain.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<style>
body, html {
  height: 100%;
  margin: 0;
}

.bg {
  /* The image used */
  background-image: url("11.jpg");

  /* Full height */
  height: 100%;

  /* Center and scale the image nicely */
  background-position: center;
  background-repeat: no-repeat;
  background-size: cover;
}

.dropbtn {
  background-color: #4CAF50;
  color: white;
  padding: 16px;
  font-size: 16px;
  border: none;
}

.dropdown {
  position: relative;
  display: inline-block;
}

.dropdown-content {
  display: none;
  position: absolute;
  background-color: #f1f1f1;
  min-width: 160px;
  box-shadow: 0px 8px 16px 0px rgba(0,0,0,0.2);
  z-index: 1;
}
```

```
.dropdown-content a {
  color: black;
  padding: 12px 16px;
  text-decoration: none;
  display: block;
}

.dropdown-content a:hover {background-color: #ddd;}

.dropdown:hover .dropdown-content {display: block;}

.dropdown:hover .dropbtn {background-color: #3e8e41;}
</style>

<meta charset="ISO-8859-1">
<title>Encryption Page</title>
</head>
<body>
<div class="bg"><center>
<div class="dropdown">
  <button class="dropbtn">Dropdown</button>
  <div class="dropdown-content">
    <a href="rsa.jsp">RSA</a>
    <a href="#">AES</a>
  </div>
</div></center>

</div>
</body>
</html>
```

FileServer.jsp

```
package shobh;


import java.io.DataInputStream;

import java.io.FileOutputStream;

import java.io.IOException;

import java.net.ServerSocket;

import java.net.Socket;


public class FileServer extends Thread {


	private ServerSocket ss;


	public FileServer(int port) {
```

```
		try {
			ss = new ServerSocket(port);
		} catch (IOException e) {
			e.printStackTrace();
		}
	}

	public run() {
		while (true) {
			try {
				Socket clientSock = ss.accept();
				saveFile(clientSock);
			} catch (IOException e) {
				e.printStackTrace();
			}
		}
	}

	private saveFile(Socket clientSock) throws IOException {
		DataInputStream dis = new DataInputStream(clientSock.getInputStream());
		FileOutputStream fos = new FileOutputStream("E:\\New folder
(2)\\server\\1.txt");
		byte[] buffer = new byte[4096];

		int filesize = 15123; // Send file size in separate msg
		int read = 0;
		int totalRead = 0;
		int remaining = filesize;
		while((read = dis.read(buffer, 0, Math.min(buffer.length, remaining))) >
0) {
			totalRead += read;
			remaining -= read;
```

```
                System.out.println("read " + totalRead + " bytes.");
                fos.write(buffer, 0, read);
            }

            fos.close();
            dis.close();
      }}
```

Decryptor.java

```
package shobh;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;



public class Decryptor
{
      private static Decryptor decrypter = new Decryptor();

      private static boolean deleteOriginal;

      private Decryptor()
      {
      }

      public static Decryptor getDecrypter(boolean originalFileDeleted)
      {
            deleteOriginal = originalFileDeleted;
```

```
	return decrypter;
}

public void decrypt(File src, File dst)
{
	if (!dst.exists())
		dst.mkdir();
	if (!dst.isDirectory())
		return;

	try
	{
		if (!src.isDirectory())
		{
			copyDecrypted(src, dst);
		} else
		{
			File[] files = src.listFiles();

			System.out.println("Decryting...");

			for (File f : files)
			{
				copyDecrypted(f, dst);
				if(deleteOriginal) f.delete();
			}

			System.out.println(files.length + " files are decrytped");
		}
	} catch (IOException e)
```

```
		{
			e.printStackTrace();
		}
	}

	public void copyDecrypted(File source, File dest) throws IOException
	{
		InputStream is = null;
		OutputStream os = null;

		try
		{
			is = new FileInputStream(source);

			byte[] buffer = new byte[1024];

			byte[] name = new byte[is.read() * 2];
			is.read(name);
			String fileName = bytesToString(name);

			os = new
FileOutputStream(dest.getPath().concat("/").concat(fileName));

			int length;

			while ((length = is.read(buffer)) > 0)
			{
				decryptBytes(buffer);
				os.write(buffer, 0, length);
			}

		} finally
```

```
		{
			is.close();
			os.close();
		}
	}

	public String bytesToString(byte[] data)
	{
		StringBuilder res = new StringBuilder();

		for (int i = 0; i < data.length / 2; i++)
		{
			char c = (char) ((data[i * 2] << 8) | data[i * 2 + 1]);
			res.append(c);
		}

		return res.toString();
	}

	private void decryptBytes(byte[] data) // Decryption Algorithm is written into
here
	{
		for (int i = 0; i < data.length; i++)
		{
			data[i] = (byte) ~data[i];
		}
	}

	public void copy(File source, File dest) throws IOException
	{
		InputStream is = null;
		OutputStream os = null;
```

```
            try
            {
                  dest = new
File(dest.getPath().concat("/").concat(source.getName()));

                  is = new FileInputStream(source);
                  os = new FileOutputStream(dest);

                  byte[] buffer = new byte[1024];

                  int length;
                  int tl = 0;

                  while ((length = is.read(buffer)) > 0)
                  {
                        tl += length;
                        os.write(buffer, 0, length);
                  }

                  System.out.println(tl + " bytes");
            } finally
            {
                  is.close();
                  os.close();
            }
      }
}
```

AESFileDecryption.java

```
package shobh;
```

```
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.security.spec.KeySpec;

import javax.crypto.Cipher;
import javax.crypto.SecretKey;
import javax.crypto.SecretKeyFactory;
import javax.crypto.spec.IvParameterSpec;
import javax.crypto.spec.PBEKeySpec;
import javax.crypto.spec.SecretKeySpec;

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

		String password = "javapapers";

		// reading the salt
		// user should have secure mechanism to transfer the
		// salt, iv and password to the recipient
		FileInputStream saltFis = new FileInputStream("E:\\min\\salt.enc");
		byte[] salt = new byte[8];
		saltFis.read(salt);
		saltFis.close();

		// reading the iv
		FileInputStream ivFis = new FileInputStream("E:\\min\\iv.enc");
		byte[] iv = new byte[16];
		ivFis.read(iv);
		ivFis.close();

		SecretKeyFactory factory = SecretKeyFactory
```

```
                        .getInstance("PBKDF2WithHmacSHA1");
                KeySpec keySpec = new PBEKeySpec(password.toCharArray(), salt, 65536,
                        256);
                SecretKey tmp = factory.generateSecret(keySpec);
                SecretKey secret = new SecretKeySpec(tmp.getEncoded(), "AES");

                // file decryption
                Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5Padding");
                cipher.init(Cipher.DECRYPT_MODE, secret, new IvParameterSpec(iv));
                FileInputStream fis = new
FileInputStream("E:\\min\\aes\\encrypted\\encr.des");
                FileOutputStream fos = new
FileOutputStream("E:\\min\\aes\\decrypted\\decrypted.txt");
                byte[] in = new byte[64];
                int read;
                while ((read = fis.read(in)) != -1) {
                      byte[] output = cipher.update(in, 0, read);
                      if (output != null)
                            fos.write(output);
                }

                byte[] output = cipher.doFinal();
                if (output != null)
                      fos.write(output);
                fis.close();
                fos.flush();
                fos.close();
                System.out.println("File Decrypted.");
      }
}
```