**Practical: 05**

**AIM:** Bob is going to send his encrypted file using public key shared by Alice using Public key infrastructure. Alice will decrypt the file by using her private key and ensure the confidentiality. Implement the following scenario using RSA algorithm.

****

After applying RSA, analyze the processing power of the computer and speed with respect to time. Try using 1024 bits of key. Discuss the issues with this scenario.

**CODE:**

**RSA Server :**

import java.net.\*;

import java.io.\*;hy

import java.security.\*;

import java.util.Base64;

import javax.crypto.Cipher;

public class RSAServer {

  private static final *String* PUBLIC\_KEY\_FILE = "publicKey.txt";

  private static final *String* PRIVATE\_KEY\_FILE = "privateKey.txt";

  private static *PrivateKey* privateKey;

  private static *PublicKey* publicKey;

  public static *void* main(*String*[] *args*) {

    try {

*KeyPairGenerator* keyGen = KeyPairGenerator.getInstance("RSA");

      keyGen.initialize(2048);

*KeyPair* keyPair = keyGen.generateKeyPair();

      publicKey = keyPair.getPublic();

      privateKey = keyPair.getPrivate();

      saveKeyToFile(PUBLIC\_KEY\_FILE, publicKey.getEncoded());

      saveKeyToFile(PRIVATE\_KEY\_FILE, privateKey.getEncoded());

*ServerSocket* serverSocket = new ServerSocket(12345);

      System.out.println("Server listening on port 12345...");

      while (true) {

*Socket* socket = serverSocket.accept();

        System.out.println("Client connected!");

        try (*BufferedWriter* writer = new BufferedWriter(new OutputStreamWriter(socket.getOutputStream()));

*BufferedReader* reader = new BufferedReader(new InputStreamReader(socket.getInputStream()))) {

          writer.write(Base64.getEncoder().encodeToString(publicKey.getEncoded()));

          writer.newLine();

          writer.flush();

*String* encryptedDataBase64 = reader.readLine();

          if (encryptedDataBase64 != null) {

*byte*[] encryptedData = Base64.getDecoder().decode(encryptedDataBase64);

            System.out.println("Received Encrypted Data: " + encryptedDataBase64);

*String* decryptedData = decrypt(encryptedData, privateKey);

            System.out.println("Decrypted Data: " + decryptedData);

          }

        }

        socket.close();

      }

    } catch (*Exception* *e*) {

      e.printStackTrace();

    }

  }

  public static *String* decrypt(*byte*[] *data*, *PrivateKey* *privateKey*) throws *Exception* {

*Cipher* cipher = Cipher.getInstance("RSA");

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

*byte*[] decryptedBytes = cipher.doFinal(data);

    return new String(decryptedBytes);

  }

  private static *void* saveKeyToFile(*String* *filename*, *byte*[] *key*) throws *IOException* {

    try (*BufferedWriter* writer = new BufferedWriter(new FileWriter(filename))) {

      writer.write(Base64.getEncoder().encodeToString(key));

    }

  }

}

**RSA Client:**

import java.net.\*;

import java.io.\*;

import java.security.\*;

import java.security.spec.X509EncodedKeySpec;

import java.util.Base64;

import javax.crypto.Cipher;

public class RSAClient {

  public static *void* main(*String*[] *args*) {

    try {

*Socket* socket = new Socket("172.25.171.217", 12345);

      System.out.println("Connected to server!");

      try (*BufferedReader* reader = new BufferedReader(new InputStreamReader(socket.getInputStream()));

*BufferedWriter* writer = new BufferedWriter(new OutputStreamWriter(socket.getOutputStream()))) {

*String* encodedPublicKey = reader.readLine();

*byte*[] decodedPublicKey = Base64.getDecoder().decode(encodedPublicKey);

*KeyFactory* keyFactory = KeyFactory.getInstance("RSA");

*PublicKey* publicKey = keyFactory.generatePublic(new X509EncodedKeySpec(decodedPublicKey));

        System.out.println("Received Public Key from Server!");

*String* message = "This is a secret document to be encrypted.";

*byte*[] encryptedData = encrypt(message, publicKey);

*String* encryptedDataBase64 = Base64.getEncoder().encodeToString(encryptedData);

        writer.write(encryptedDataBase64);

        writer.newLine();

        writer.flush();

        System.out.println("Encrypted data sent to server!");

      }

      socket.close();

    } catch (*Exception* *e*) {

      e.printStackTrace();

    }

  }

  public static *byte*[] encrypt(*String* *data*, *PublicKey* *publicKey*) throws *Exception* {

*Cipher* cipher = Cipher.getInstance("RSA");

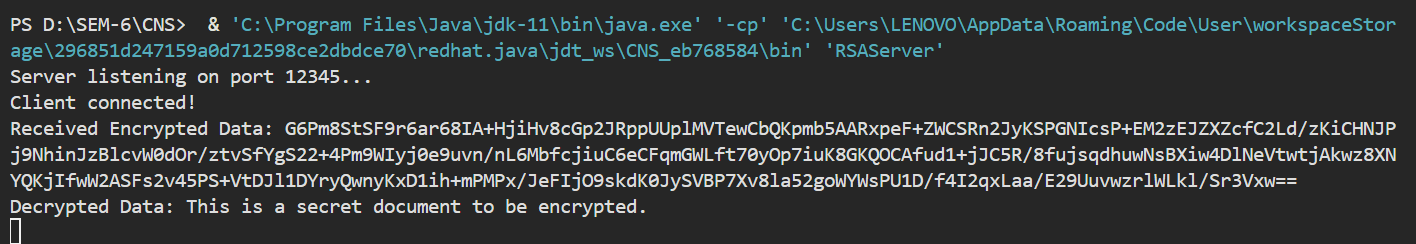
    cipher.init(Cipher.ENCRYPT\_MODE, publicKey);

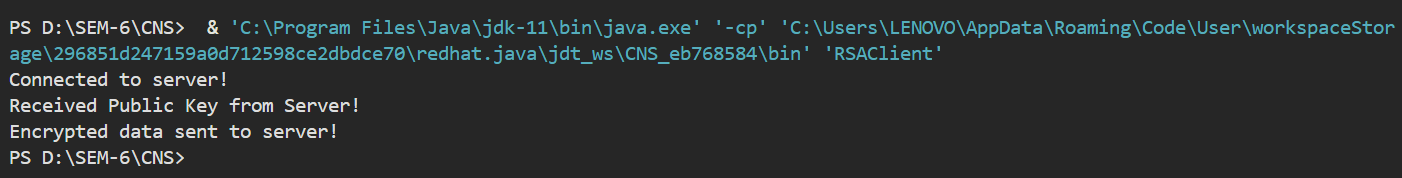
    return cipher.doFinal(data.getBytes());

  }

}

**OUTPUT:**

****

****