**Name:** Hung Viet Luu

**Link to Project:** <https://github.com/hvluu/CS380/tree/master/Projects/Project7>

**Instructions: do not compile the file in IDE, instead compiling it in using terminal (or cmd in Windows). After that, input the following commands**

1. **java FileTransfer makekeys //this has to be executed first to make the private and public keys**
2. **java FileTransfer server <private key file> <port number> //this is to specify if the machine running as the server**
3. **java FileTransfer client <public key file> <host IP address> <port number> //this is to specify if the machine running as the client and connect to the server using the same port**

**FileTransfer.java**

*import java.security.\*;*

*import java.security.interfaces.\*;*

*import java.io.\*;*

*import java.net.\*;*

*import java.nio.file.\*;*

*import java.util.\*;*

*import java.util.zip.CRC32;*

*import javax.crypto.\*;*

*public class FileTransfer {*

*//encrypted file transfer system.*

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

*boolean modeSelected = false;*

*if(args.length >= 1){*

*if(args[0].equals("makekeys") && args.length == 1){*

*makeKeys();*

*modeSelected = true;*

*} else if(modeSelected){*

*System.out.println("Proper usage to make keys: java FileTransfer makekeys");*

*}*

*if(args[0].equals("client") && args.length == 4 ){*

*String pubKey = args[1];*

*String host = args[2];*

*String port = args[3];*

*clientMode(port, host, pubKey);*

*modeSelected = true;*

*} else if(args[0].equals("client")){*

*System.out.println("Proper usage to run in client mode: java FileTransfer client <public key file> <host> <port>");*

*}*

*if(args[0].equals("server") && args.length == 3){*

*String privKey = args[1];*

*String port = args[2];*

*serverMode(privKey, port);*

*modeSelected = true;*

*} else if(args[0].equals("server")){*

*System.out.println("Proper usage to run in server mode: java FileTransfer server <private key file> <port>");*

*}*

*} else{*

*System.out.println("Proper usage requires at least one command line argument.");*

*System.out.println("Try one of the following options to see details on how to use: ");*

*System.out.println("'java FileTransfer makekeys' this will run without any addional arguments");*

*System.out.println("'java FileTransfer client' requires additional arguments");*

*System.out.println("'java FileTransfer server' requires additional arguments");*

*}*

*}*

*//acts as a server receiving the files*

*private static void serverMode(String privKey, String port) {*

*try {*

*boolean receiving = true;*

*while(receiving){*

*//connections and IO streams*

*ServerSocket serSocket = new ServerSocket(Integer.parseInt(port));*

*Socket socket = serSocket.accept();*

*InputStream is = socket.getInputStream();*

*ObjectInputStream ois = new ObjectInputStream(is);*

*//starting and creating the the sessions key*

*StartMessage s = (StartMessage)ois.readObject();*

*byte[] wrappedSessionKey = s.getEncryptedKey();*

*ObjectInputStream in = new ObjectInputStream(new FileInputStream(privKey));*

*RSAPrivateKey rsa = (RSAPrivateKey) in.readObject();*

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

*cipher.init(Cipher.UNWRAP\_MODE, rsa);*

*Key key = cipher.unwrap(wrappedSessionKey, "AES", Cipher.SECRET\_KEY);*

*System.out.println(s.getFile());*

*OutputStream os = socket.getOutputStream();*

*ObjectOutputStream oos = new ObjectOutputStream(os);*

*int seqNum = 0;*

*AckMessage ack = new AckMessage(seqNum);*

*seqNum++;*

*oos.writeObject(ack);*

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

*cipher.init(Cipher.DECRYPT\_MODE, key);*

*String message = "";*

*String fileName = s.getFile().replace(".txt", "2.txt");*

*PrintWriter pw = new PrintWriter(fileName, "UTF-8");*

*byte[] b = null;*

*byte[] decrypted = null;*

*Message m;*

*CRC32 crc = new CRC32();*

*int numChunks = (int) s.getSize()/s.getChunkSize();*

*if(s.getSize() % s.getChunkSize() >= 1){*

*numChunks++;*

*}*

*//receiving the files and sending acknowledgements*

*for(int i = 0; i < numChunks; i++){*

*m = (Message)ois.readObject();*

*b = ((Chunk)m).getData();*

*decrypted = cipher.doFinal(b);*

*crc.update(decrypted);*

*int checkSum = (int)crc.getValue();*

*crc.reset();*

*if(checkSum != ((Chunk)m).getCrc()){*

*System.out.println("there was an error when sending the data");*

*}*

*System.out.println("Chunk recieved: [" + seqNum + "/" + numChunks + "]");*

*message += new String(decrypted);*

*seqNum++;*

*ack = new AckMessage(seqNum);*

*oos.writeObject(ack);*

*}*

*System.out.println("Transfer complete. Created file: " + fileName);*

*pw.print(message);*

*pw.close();*

*ois.close();*

*oos.close();*

*in.close();*

*is.close();*

*os.close();*

*serSocket.close();*

*socket.close();*

*}*

*} catch (Exception e) {*

*System.out.println("error");*

*e.printStackTrace();*

*}*

*}*

*//acts as a client sending a file*

*private static void clientMode(String port, String host, String pubKey) throws Exception {*

*boolean sendingFiles = true;*

*while(sendingFiles){*

*try (Socket socket = new Socket(host, Integer.parseInt(port))) {*

*System.out.println("Connected to: " + host + " on port: " + port);*

*//Read the public key, and create the session key*

*ObjectInputStream in = new ObjectInputStream(new FileInputStream(pubKey));*

*RSAPublicKey publicKey = (RSAPublicKey) in.readObject();*

*KeyGenerator keyGen = KeyGenerator.getInstance("AES");*

*keyGen.init(128);*

*SecretKey sessionKey = keyGen.generateKey();*

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

*cipher.init(Cipher.WRAP\_MODE, publicKey);*

*byte[] wrappedSessionKey = cipher.wrap(sessionKey);*

*//Get the file to be sent from the user*

*System.out.print("Enter the file path: " );*

*Scanner kb = new Scanner(System.in);*

*String filePath = kb.next();*

*Path path = Paths.get(filePath);*

*byte[] data = Files.readAllBytes(path);*

*//get the chunk size from the user*

*System.out.print("Enter chunk size [1024]: ");*

*int chunkSize = kb.nextInt();*

*//begin*

*StartMessage sm = new StartMessage(path.getFileName().toString(), wrappedSessionKey, chunkSize);*

*System.out.println("Sending: " + filePath + "\tFile Size: " + data.length);*

*//I/O streams*

*OutputStream os = socket.getOutputStream();*

*ObjectOutputStream oos = new ObjectOutputStream(os);*

*oos.writeObject(sm);*

*InputStream is = socket.getInputStream();*

*ObjectInputStream ois = new ObjectInputStream(is);*

*//get number of chunks, and the leftovers after initial chunks are sent*

*int leftOver = data.length % chunkSize;*

*int chunks = data.length / chunkSize;*

*int totalChunks = chunks;*

*if(leftOver > 0){*

*totalChunks++;*

*}*

*System.out.println("Sending " + totalChunks + " chunks");*

*int placement = 0;*

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

*cipher.init(Cipher.ENCRYPT\_MODE, sessionKey);*

*CRC32 crc = new CRC32();*

*for(int i = 0; i < chunks; i++){*

*//package a chunk*

*byte[] toSend = new byte[chunkSize];*

*byte[] encrypted = null;*

*for(int j = 0, k = placement; j < chunkSize; j++, k++){*

*toSend[j] = data[k];*

*}*

*//get everything ready to send - seqNum, checksum, and encrypted data*

*AckMessage ack = (AckMessage)ois.readObject();*

*crc.update(toSend);*

*int checkSum = (int)crc.getValue();*

*crc.reset();*

*encrypted = cipher.doFinal(toSend);*

*//send the chunk*

*Chunk chunk = new Chunk(ack.getSeq(), encrypted, checkSum);*

*oos.writeObject(chunk);*

*System.out.println("Chunks completed " + "[" + (i+1) + "/" + totalChunks + "]");*

*placement += chunkSize;*

*}*

*if(leftOver > 0){*

*byte[] toSend = new byte[leftOver];*

*byte[] encrypted = null;*

*for(int j = 0, k = placement; j < leftOver; j++, k++){*

*toSend[j] = data[k];*

*}*

*AckMessage ack = (AckMessage)ois.readObject();*

*crc.update(toSend);*

*int checkSum = (int)crc.getValue();*

*crc.reset();*

*encrypted = cipher.doFinal(toSend);*

*Chunk chunk = new Chunk(ack.getSeq(), encrypted, checkSum);*

*System.out.println("Chunks completed " + "[" + totalChunks + "/" + totalChunks + "]");*

*oos.writeObject(chunk);*

*}*

*System.out.println("Would you like to: ");*

*System.out.println("Enter '1' to send another file.");*

*System.out.println("Enter '2' to quit");*

*int choice = kb.nextInt();*

*if(choice == 1){*

*in.close();*

*oos.close();*

*ois.close();*

*os.close();*

*is.close();*

*socket.close();*

*} else{*

*in.close();*

*oos.close();*

*ois.close();*

*sendingFiles = false;*

*}*

*}*

*}*

*}*

*//makes the public and private keys*

*private static void makeKeys() {*

*try {*

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

*gen.initialize(4096);*

*KeyPair keyPair = gen.genKeyPair();*

*PrivateKey privateKey = keyPair.getPrivate();*

*PublicKey publicKey = keyPair.getPublic();*

*try (ObjectOutputStream oos = new ObjectOutputStream(*

*new FileOutputStream(new File("public.bin")))) {*

*oos.writeObject(publicKey);*

*}*

*try (ObjectOutputStream oos = new ObjectOutputStream(*

*new FileOutputStream(new File("private.bin")))) {*

*oos.writeObject(privateKey);*

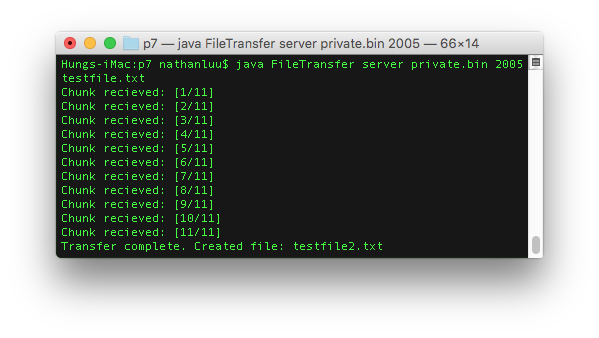
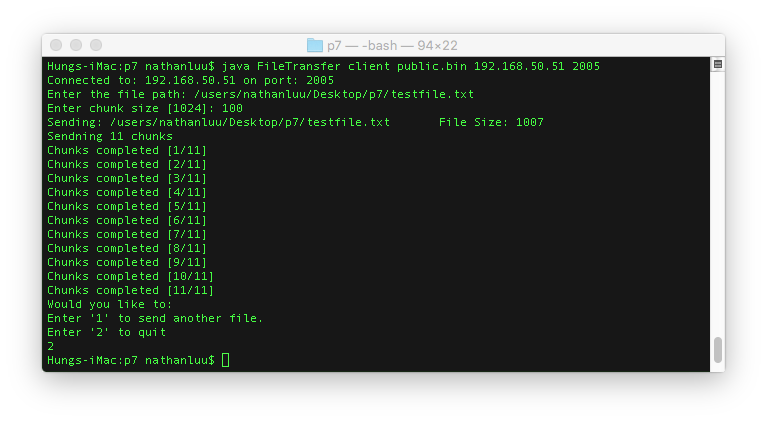
*}*

*} catch (NoSuchAlgorithmException | IOException e) {*

*e.printStackTrace(System.err);*

*}*

*}*

*}*