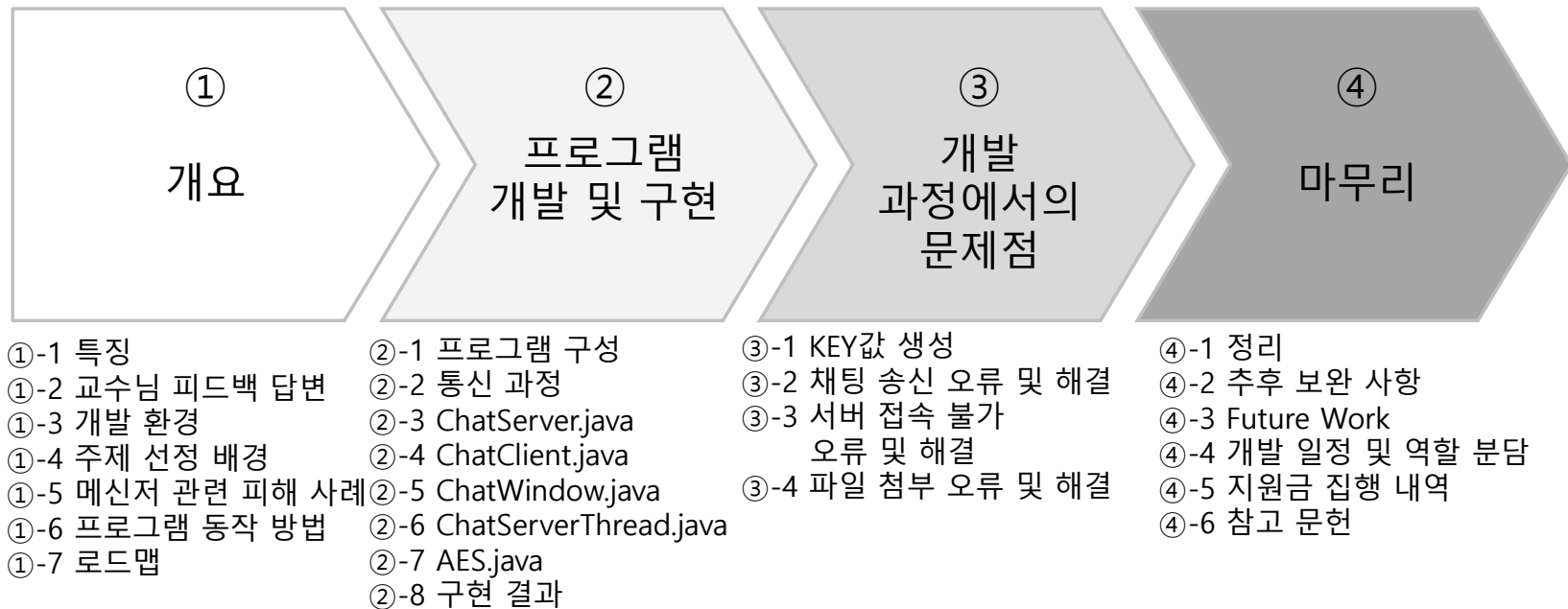


양방향 통신 보안 메신저 최종발표

지도교수 : 이후진 교수님
팀장 : 1771469 이광현
조원 : 1094017 배재현
조원 : 1771075 나관우

목차



① 개요

①-1 특징

기본적인 메신저의 기능에 더불어

메시지 블록암호화(AES-128,256 or LEA-128,256)를 통해 중간자 공격이 불가능하게 하고

대화가 끝남과 동시에 채팅기록, 로그를 모두 삭제하여

해킹으로 인한 사생활 누출, 사회 공학 공격 등을 불가능하게 만든

비밀 보안 메신저

①-2 교수님 피드백 답변

1. 중간자 공격을 막기 위한 메신저라고 말하고 있지만, 도청 (eavesdropping)을 막기위한 메신저로 보임. 중간자 공격과 도청의 개념을 정리할 필요가 있어 보임.
2. 서버가 비밀키를 클라이언트한테 단순히 전달하는데, 이 부분을 개선할 필요가 있어보임.
3. 취약점 테스트를 계획하고 있는데 이 때 공격자에 대한 정의를 분명히 할 필요가 있어 보임. 예를 들어, 네트워크 레벨에서 중간자 공격 또는 도청을 하는 공격자 등.



1. 도청을 방지할 수 있도록 블록암호화 알고리즘인 AES-128, 256와 LEA-128, 256을 선택할 수 있도록 구현하였습니다.
2. 서버가 비밀키를 전달하지 않고, 서버 IP를 이용해 비밀키를 생성하도록 하였습니다.
3. 취약점 테스트는 진행하지 못했습니다. 향후 프로젝트 계획에 추가하여 완성도가 더욱 높은 프로그램을 만들도록 노력하겠습니다.

①-3 개발 환경

운영체제 : Window 10 Home Edition x64

CPU : Intel® Core(TM) i5-8400

개발 환경 : JDK-13.0.2, Eclipse IDE for eclipse Committers

컴퓨터에 대한 기본 정보 보기

Windows 버전

Windows 10 Home

© 2019 Microsoft Corporation. All rights reserved.



시스템

프로세서: Intel(R) Core(TM) i5-8400 CPU @ 2.80GHz 2.80 GHz
설치된 메모리(RAM): 16.0GB(15.9GB 사용 가능)
시스템 종류: 64비트 운영 체제, x64 기반 프로세서
펜 및 터치: 이 디스플레이에 사용할 수 있는 펜 또는 터치식 입력이 없습니다.

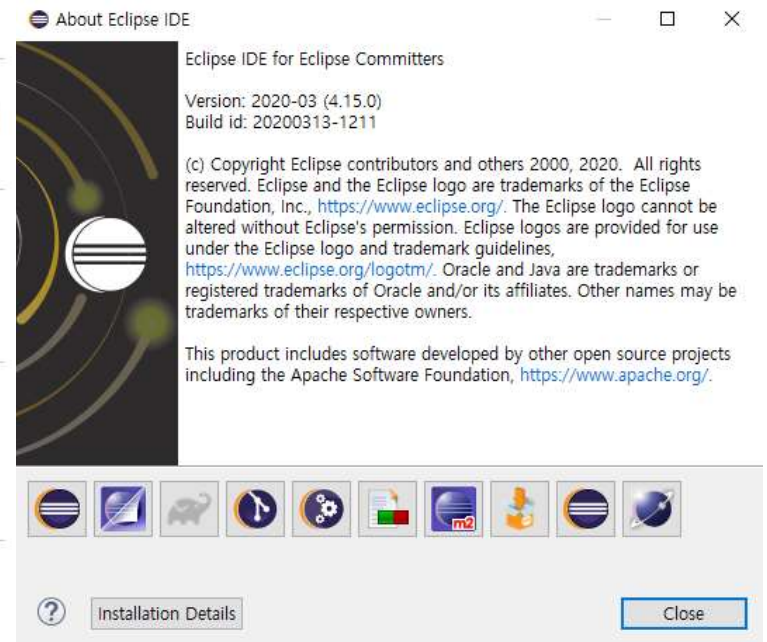
컴퓨터 이름, 도메인 및 작업 그룹 설정

컴퓨터 이름:
전체 컴퓨터 이름:
컴퓨터 설명:
작업 그룹:

Windows 제품 인증

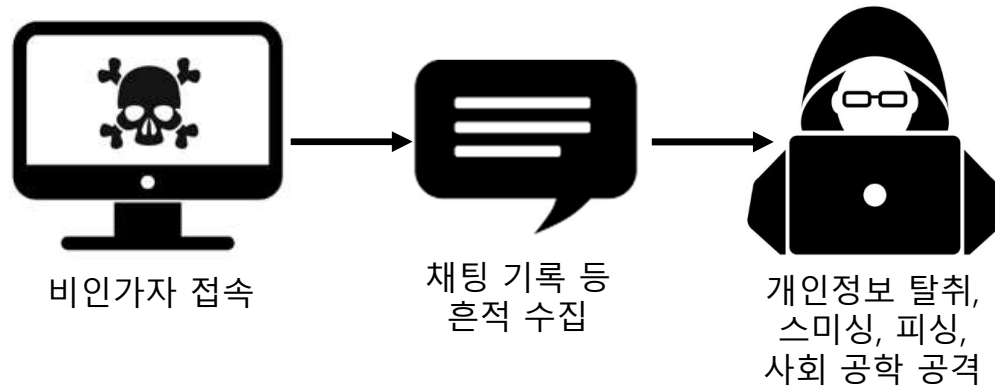
Windows 제품 인증을 받았습니다. [Microsoft 소프트웨어 사용 조건 읽기](#)

제품 ID:



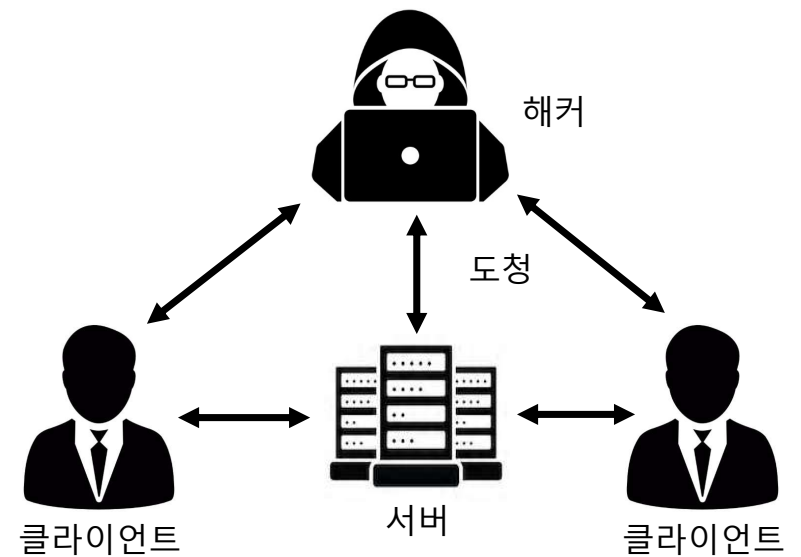
①-4 주제 선정 배경

컴퓨터 해킹으로 인한 개인정보 탈취



→ 채팅 기록 및 로그
영구 삭제로 흔적 제거

도청을 통한
메신저 내용 및 개인정보 탈취



→ 블록암호화 적용으로
채팅 내용 도청 방지

①-5 메신저 관련 피해 사례



2009년 12월 12월 구글 해킹 당시 전세계 수백만 명의 구글 웹서비스 이용을 관리하는 암호 체계가 침입자들에게 뚫렸다고 뉴스에 나온 적이 있다.

사건의 발단은 구글 직원이 사용하던 MS메신저를 통해 접근이 되었으며, 개발자 팀이 사용하는 소프트웨어 저장공간의 통제권까지 모두 접근할 수 있었다.

이들이 공격한 암호체계는 '가이아'라고 불리고 있으며 이 프로그램은 인터넷 사용자나 구글 직원들이 이메일이나 비즈니스 앱과 같은 서비스를 이용할 때 패스워드를 입력하도록 하는 소프트웨어다.

가이아는 구글 지메일(Gmail) 등의 가입자가 패스워드를 입력하고 로그인을 하면 이를 인증해 가입한 모든 서비스를 원스톱으로 이용할 수 있게 해주는 기능을 갖고 있다. 패스워드를 사용하는 구글의 거의 모든 서비스와 연결되어 있기 때문에 가이아 시스템이 해킹 당할 경우 구글의 모든 정보가 전부 노출될 위험이 있다.

①-5 메신저 관련 피해 사례

카톡보다 안전하다는 메신저 '텔레그램'도 해킹 당했다

강동철 기자

입력 2016.08.03 23:07

한때 한국에서 '메신저 망명(亡命)'이란 신조어를 만들어냈던 독일의 모바일 메신저인 '텔레그램(telegram)'이 이란에서 해킹 공격을 받아 대규모 피해를 입었다고 로이터가 3일(현지 시각) 보도했다.



Telegram

2016년 8월 독일의 모바일 메신저인 '텔레그램(telegram)'이 이란에서 해킹 공격을 받아 대규모 피해를 입었다고 로이터 통신이 보도한 적이 있다.

이번 해킹 사건은 이란 해커들이 저질렀고, 텔레그램 가입자 1500만 명의 전화번호와 일부 이용자의 대화 내용이 유출됐다. 이에 2000만명이 텔레그램을 사용하는데 이용자의 정보가 유출되었다고 한다.

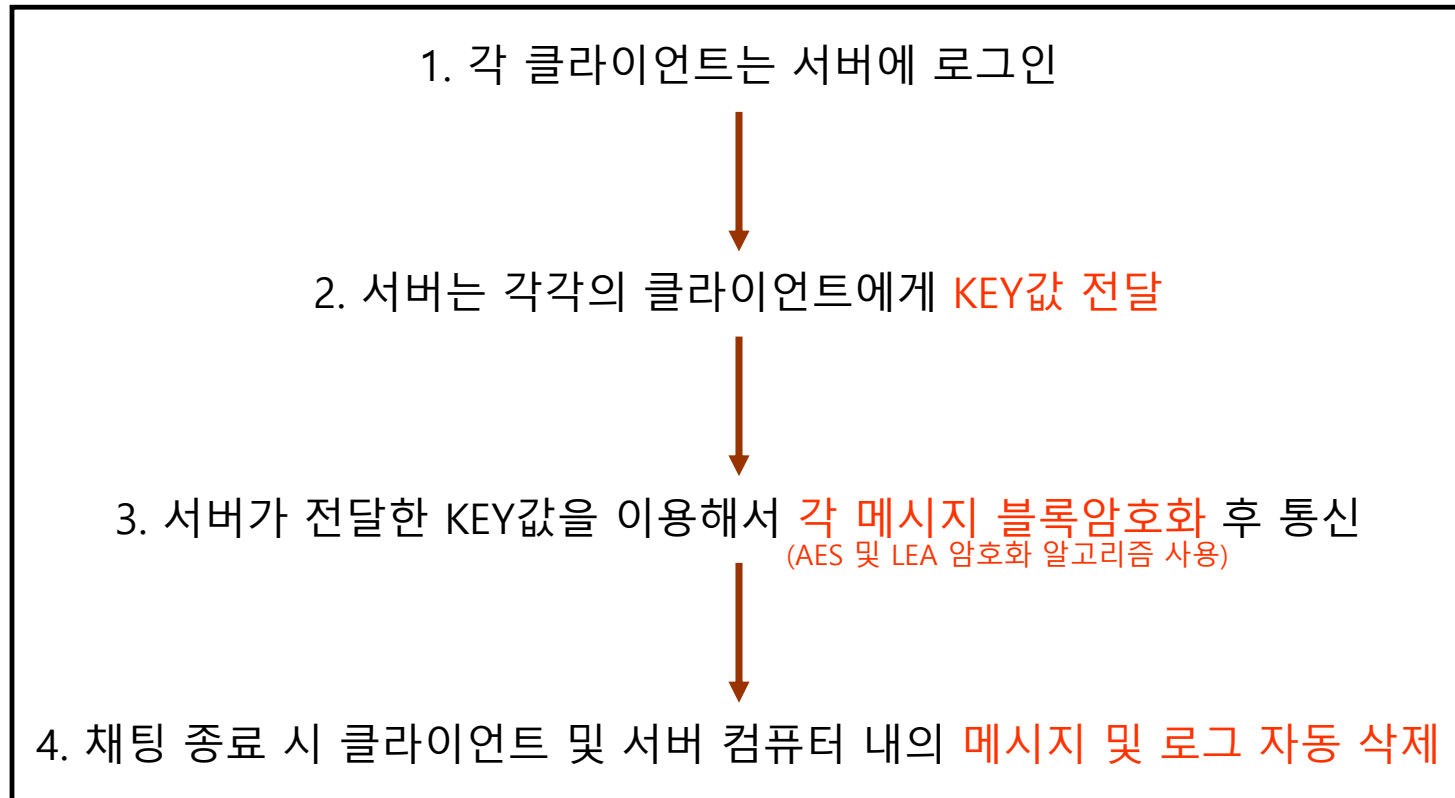
텔레그램은 메신저 대화 내용을 자사의 서버에 저장하지 않고 전달만 하는 방식이기 때문에 감청이나 해킹으로부터 안전하다고 알려졌다.

이에 2014년 국내 수사기관이 카카오톡 대화 내용을 감청한다는 소식이 전해지자 100만명이 넘는 카카오톡 이용자가 텔레그램으로 이동하는 '메신저 망명' 사태가 벌어지기도 했다.

그런데 텔레그램이 더 이상 해킹의 안전지대가 아니라는 사실이 드러난 것이다.

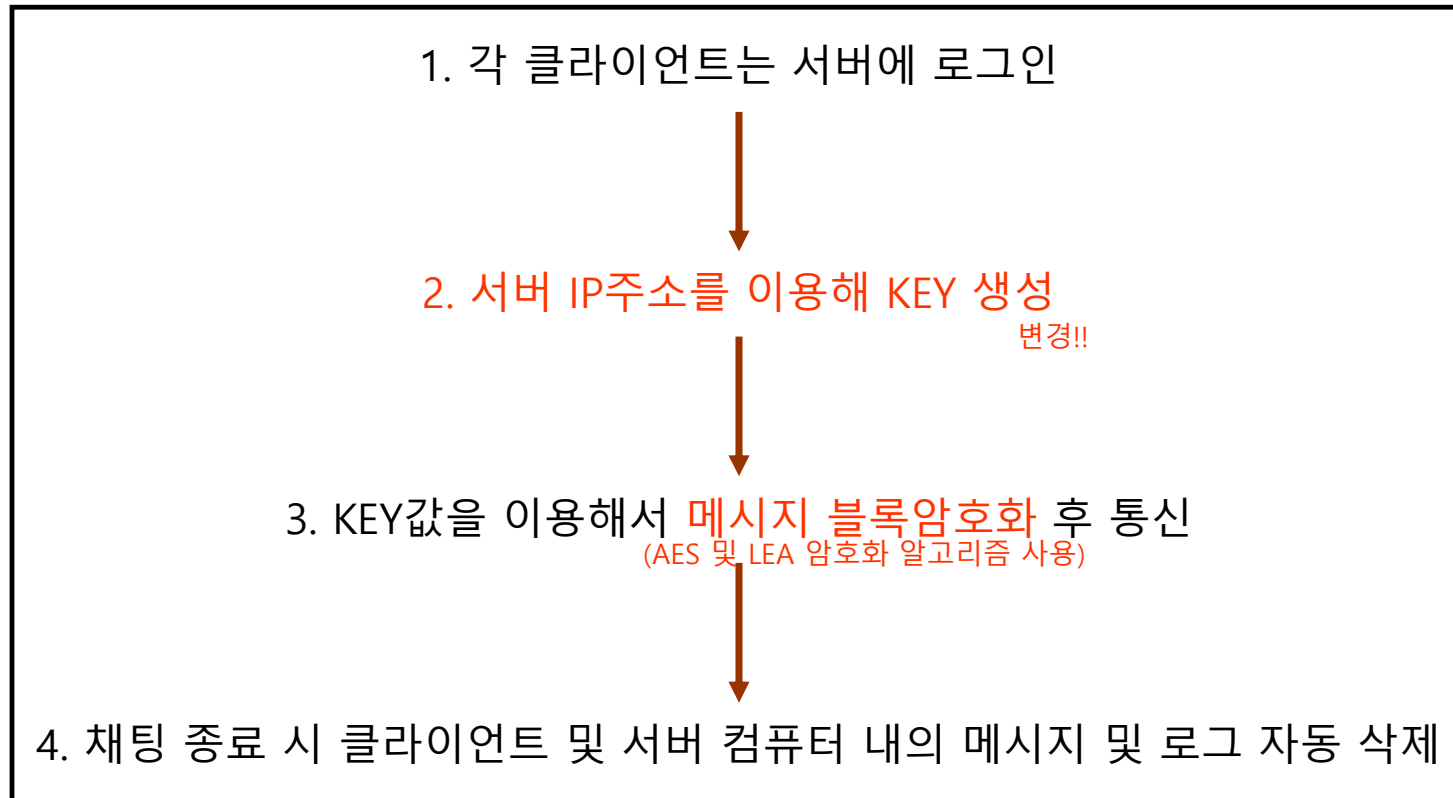
①-6 프로그램 동작 방법

초기 계획된 동작 방법

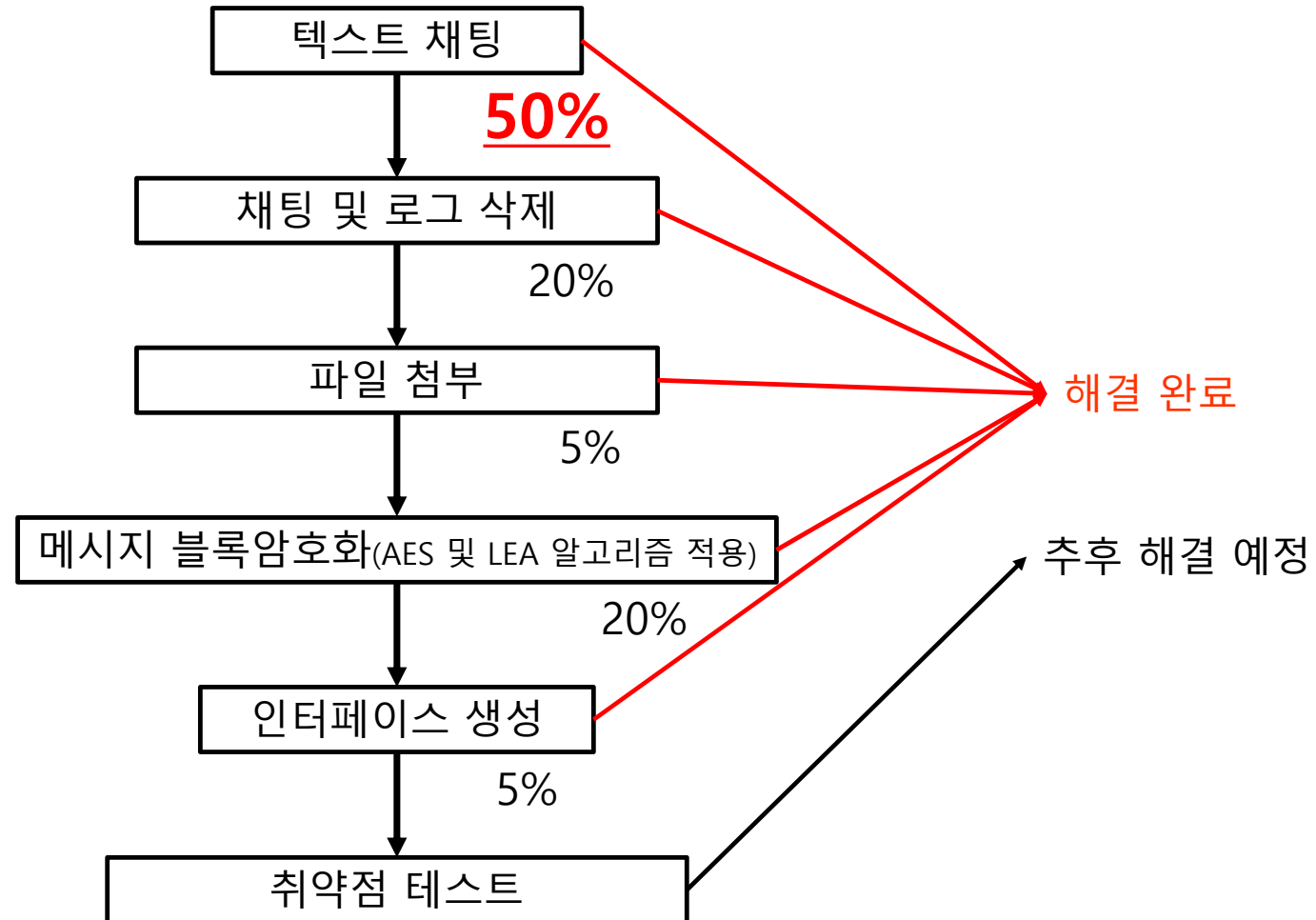


①-6 프로그램 동작 방법

현재 동작 방법

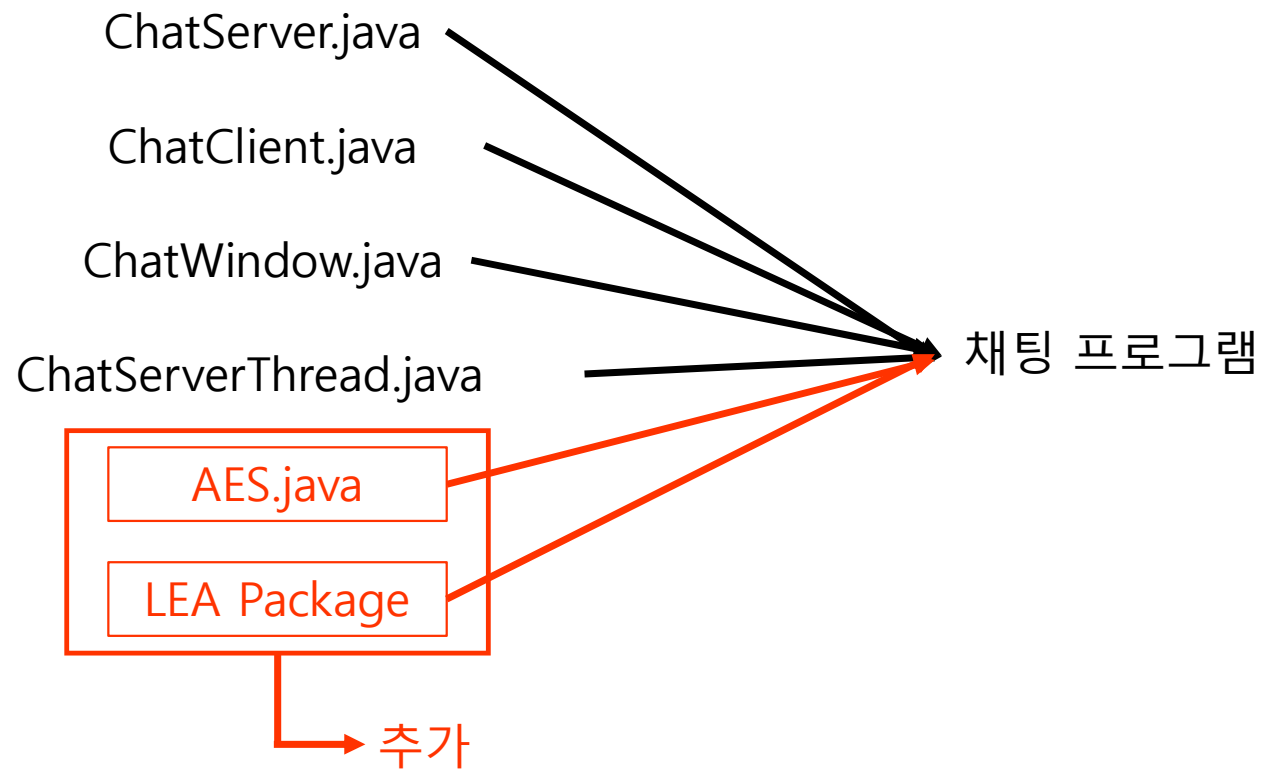


①-7 로드맵



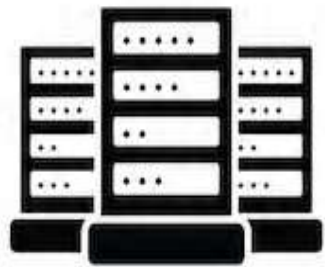
② 프로그램 개발 및 구현

②-1 프로그램 구성



②-2 통신 과정

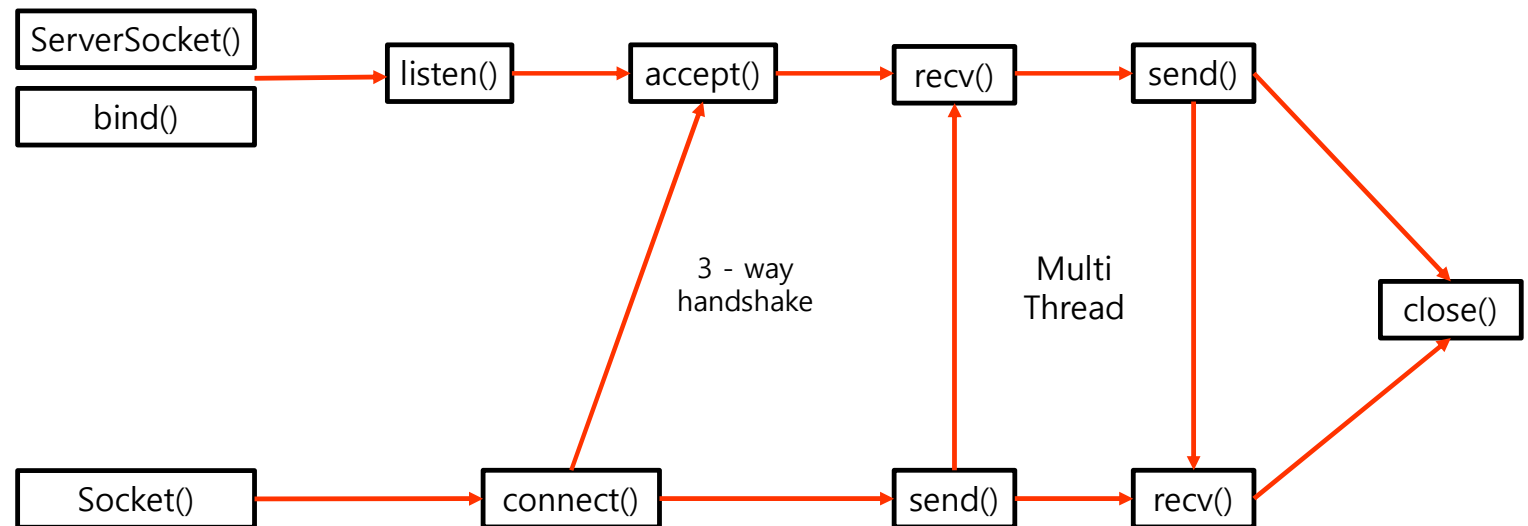
3-way handshake로 작동



Server



Client



②-3 ChatServer.java

```
serverSocket = new ServerSocket();
```

→ 서버 소켓 생성

```
String hostAddress = InetAddress.getLocalHost().getHostAddress();  
serverSocket.bind( new InetSocketAddress(hostAddress, PORT) );  
consoleLog("연결 기다림 - " + hostAddress + ":" + PORT);
```

→ 바인딩

```
while(true)  
{  
    Socket socket = serverSocket.accept();  
    new ChatServerProcessThread(socket, listWriters).start();  
}
```

→ 클라이언트 접속 대기

②-4 ChatClient.java

```
Socket socket = new Socket();
```

클라이언트 소켓 생성

```
try  
{
```

```
    socket.connect( new InetSocketAddress(SERVER_IP, SERVER_PORT) );
```

서버에 접속

```
    consoleLog("채팅방에 입장하였습니다.");
```

```
    new ChatWindow(name, socket).show();
```

```
    PrintWriter pw = new PrintWriter(new OutputStreamWriter(socket.getOutputStream(), StandardCharsets.UTF_8), true);
```

```
    String request = "join:" + name + "\r\n";
```

```
    pw.println(request);
```

```
}
```

```
catch (IOException e)
```

```
{
```

```
    e.printStackTrace();
```

```
}
```

ChatWindow.java의
show() 메서드를 호출하여
메신저 구동 시작

②-5 ChatWindow.java

```
public ChatWindow(String name, Socket socket)
{
    this.name = name;
    frame = new Frame(name);
    pannel = new Panel();
    buttonSend = new Button("Send");
    textField = new TextField();
    textArea = new TextArea(30, 80);
    this.socket = socket;

    new ChatClientReceiveThread(socket).start();
}
```

```
public void show()
{
    // Button
    buttonSend.setBackground(Color.GRAY);
    buttonSend.setForeground(Color.WHITE);
    buttonSend.addActionListener( new ActionListener()
    {
        @Override
        public void actionPerformed( ActionEvent actionEvent )
        {
            sendMessage();
        }
    });
}
```

ChatClientApp.java에서 호출된
ChatWindow.java와 show() 메서드

```
private class ChatClientReceiveThread extends Thread
{
    Socket socket = null;

    ChatClientReceiveThread(Socket socket)
    {
        this.socket = socket;
    }

    public void run()
    {
        try
        {
            BufferedReader br =
                new BufferedReader(new InputStreamReader(socket.getInputStream(), StandardCharsets.UTF_8));
            while(true)
            {
                String msg = br.readLine();
                textArea.append(msg);
                textArea.append("\n");
            }
        }
        catch (IOException e)
        {
            e.printStackTrace();
        }
    }
}
```

ChatClientReceiveThread는
서버가 전달하는 메시지를
수신하기 위한 스레드

클라이언트가 메시지를 입력하면
서버는 broadcast를 통해 모든 클라이언트에게
메시지를 송신하는데, 이 메시지를 수신하기 위한
스레드 동작 부분

②-5 ChatWindow.java

```
public void show()
{
    buttonSend.addActionListener( new ActionListener()
    {
        @Override
        public void actionPerformed( ActionEvent actionEvent )
        {
            sendMessage();
        }
    });
}
```

Send 버튼에
ActionListener 설정

```
// TextField
textField.setColumns(80);
textField.addKeyListener( new KeyAdapter()
{
    public void keyReleased(KeyEvent e)
    {
        char keyCode = e.getKeyChar();
        if (keyCode == KeyEvent.VK_ENTER)
        {
            sendMessage();
        }
    }
});
```

```
buttonAttach.addActionListener( new ActionListener()
{
    @Override
    public void actionPerformed( ActionEvent actionEvent )
    {
        sendFile();
    }
});
```

```
private void sendMessage()
{
    PrintWriter pw;
    try
    {
        pw = new PrintWriter(new OutputStreamWriter(socket.getOutputStream(), StandardCharsets.UTF_8), true);
        String message = textField.getText();
        String EncMessage = AES256.encrypt(message, ChatClientApp.SERVER_IP);
        AES128의 경우 이 부분이 AES128.encrypt(message, ChatClientApp.SERVER_IP);
        String request = "message:" + EncMessage + "\r\n";
        pw.println(request);

        textField.setText("");
        textField.requestFocus();
    }
    catch (IOException e)
    {
        e.printStackTrace();
    }
}
```

TextField에 텍스트 입력 후 엔터키를 누르거나
Send 버튼을 클릭하면 sendMessage() 메서드가
실행되어 입력한 텍스트가 서버로 전송된 후 broadcast됨.

②-5 ChatWindow.java

```
public void show()
{
    buttonSend.addActionListener( new ActionListener()
    {
        @Override
        public void actionPerformed( ActionEvent actionEvent )
        {
            sendMessage();
        }
    });

    // Textfield
    textField.setColumns(80);
    textField.addKeyListener( new KeyAdapter()
    {
        public void keyReleased(KeyEvent e)
        {
            char keyCode = e.getKeyChar();
            if (keyCode == KeyEvent.VK_ENTER)
            {
                sendMessage();
            }
        }
    });

    buttonAttach.addActionListener( new ActionListener()
    {
        @Override
        public void actionPerformed( ActionEvent actionEvent )
        {
            sendFile();
        }
    });
}
```

```
private void sendFile()
{
    System.out.println("Send File execute.");
    JFrame f = new JFrame();
    //f.setSize(400,300);
    f.setVisible(false);
    f.setLayout(null);
    FileDialog dialog = new FileDialog(f, "저장", FileDialog.LOAD);
    dialog.setVisible(true);

    String dir = dialog.getDirectory();
    String filename = dialog.getFile();
    String FilePath = dir + filename;
    //System.out.println(FilePath);

    File AttachFile = new File(FilePath);
    if(!AttachFile.exists())
    {
        System.out.println("File Doesn't Exist!!");
        return;
    }
    PrintWriter pw;
    try
    {
        pw = new PrintWriter(new OutputStreamWriter(socket.getOutputStream(), StandardCharsets.UTF_8), true);
        String request = "file:" + filename + "\r\n";
        pw.println(request);
    }
    catch(IOException e)
    {
        e.printStackTrace();
    }
}
```

FileDialog로 파일 경로를 얻은 후
파일을 열고 파일 이름을 전송

②-5 ChatWindow.java

```
public void show()
{
    buttonSend.addActionListener( new ActionListener()
    {
        @Override
        public void actionPerformed( ActionEvent actionEvent )
        {
            sendMessage();
        }
    });

    // Textfield
    textField.setColumns(80);
    textField.addKeyListener( new KeyAdapter()
    {
        public void keyReleased(KeyEvent e)
        {
            char keyCode = e.getKeyChar();
            if (keyCode == KeyEvent.VK_ENTER)
            {
                sendMessage();
            }
        }
    });

    buttonAttach.addActionListener( new ActionListener()
    {
        @Override
        public void actionPerformed( ActionEvent actionEvent )
        {
            sendFile();
        }
    });
}
```

```
try
{
    InputStream fis = new FileInputStream(FilePath);
    OutputStream os = socket.getOutputStream();
    while((readBytes = fis.read(buffer)) > 0)
    {
        os.write(buffer, 0, readBytes);
    }

    fis.close();
    os.flush();
    os.close();
    //os.close();
    //socket.shutdownOutput();
}
catch(FileNotFoundException e)
{
    System.out.println("ChatWindow : FileNotFoundException");
    e.printStackTrace();
}
catch(IOException e)
{
    System.out.println("ChatWindow : IOException");
    e.printStackTrace();
}

System.out.println("File Transfer completed.");
}
```

그 후 파일을 FileInputStream으로 열고
os.write()하여 파일 전송

②-5 ChatWindow.java

```
frame.addWindowListener(new WindowAdapter()
{
    public void windowClosing(WindowEvent e)
    {
        PrintWriter pw;
        try
        {
            pw = new PrintWriter(new OutputStreamWriter(socket.getOutputStream(), StandardCharsets.UTF_8), true);
            String request = "quit\r\n";
            pw.println(request);
            LogDel();
            System.exit(0);
        }
        catch (IOException e1)
        {
            e1.printStackTrace();
        }
    }
});
frame.setVisible(true);
frame.pack();
```

채팅창이 닫힐 때 종료 요청을
서버에 보낸 후 로그 삭제

```
private static void LogDel() throws FileNotFoundException
{
    File myfile = new File(Path);
    final long size = myfile.length();
    System.out.println("Size of File : " + size);

    PrintWriter output = new PrintWriter(Path);
    for(int i=1; i<size; i++)
    {
        double dVal = Math.random();
        char ran = (char)(dVal*255);
        output.print(ran);
    }
    output.close();

    if( myfile.exists() )
    {
        if( myfile.delete() )
        {
            System.out.println("Delete success.");
        }
        else
        {
            System.out.println("Delete Failed.");
        }
    }
    else
    {
        System.out.println("File doesn't exist!!");
    }
}
```

로그 삭제 알고리즘
로그 파일의 크기만큼
난수를 대입 후 로그파일 삭제

②-6 ChatServerThread.java

```
public ChatServerProcessThread(Socket socket, List<PrintWriter> listWriters)
{
    this.socket = socket;
    this.listWriters = listWriters;
}
```

listWriters 변수는
채팅 서버에 연결된
모든 클라이언트들을
저장하고 있는 List 임
(Join시 추가)

```
@Override
public void run()
{
    while(true)
    {
        try
        {
            BufferedReader buffereedReader =
                new BufferedReader(new InputStreamReader(socket.getInputStream(), StandardCharsets.UTF_8));

            PrintWriter printWriter =
                new PrintWriter(new OutputStreamWriter(socket.getOutputStream(), StandardCharsets.UTF_8));

            String request = buffereedReader.readLine();
```

ChatServerProcessThread.java는
ChatWindow에서 수신되는 요청인
request를 멀티스레드로 계속 수신

②-6 ChatServerThread.java

```
@Override
public void run()
{
    while(true)
    {
        try
        {
            BufferedReader buffereedReader =
                new BufferedReader(new InputStreamReader(socket.getInputStream(), StandardCharsets.UTF_8));

            PrintWriter printWriter =
                new PrintWriter(new OutputStreamWriter(socket.getOutputStream(), StandardCharsets.UTF_8));

            String request = buffereedReader.readLine();

            if(request == null)
            {
                consoleLog("클라이언트로부터 연결 끊김");
                doQuit(printWriter);
                break;
            }

            String[] tokens = request.split(":");
            if("join".equals(tokens[0]))
            {
                doJoin(tokens[1], printWriter);
            }
            else if("message".equals(tokens[0]))
            {
                doMessage(tokens[1]);
            }
            else if("quit".equals(tokens[0]))
            {
                doQuit(printWriter);
            }
            else if("file".equals(tokens[0]))
            {
                doFile(tokens[1]);
            }
        }
        catch(ArrayIndexOutOfBoundsException e)
        {
            consoleLog("0쓰기 오류");
        }
        catch(FileNotFoundException e)
        {
            System.out.println("FileNotFound");
        }
        catch(IOException e)
        {
            consoleLog(this.nickname + "님이 채팅방을 나갔습니다.");
            try
            {
                LogDel();
            }
            catch (FileNotFoundException e1)
            {
                System.out.println("Cannot Find or Delete '../.metadata/.log'");
            }
            break;
        }
    }
}
```

Chatwindow에서 서버로 보내는 요청은
PrintWriter.println(String request)하여
전송되는데,request의 :을 구분하여
만든 배열인 tokens의 tokens[0]에는
요청의 종류, Tokens[1]에는
요청에 관련된 내용 저장

클라이언트 종료 후
로그 삭제 실행
로그 삭제 알고리즘은
ChatWindow의 것과 같음

②-6 ChatServerThread.java

tokens[0]이 quit인 경우

```
private void doQuit(PrintWriter writer)
{
    removeWriter(writer);

    String data = this.nickname + "님이 퇴장했습니다.";
    try
    {
        LogDel();
    }
    catch (FileNotFoundException e1)
    {
        System.out.println("Cannot Find or Delete '../metadata/.log'");
    }
    broadcast(data);
}
```

```
private void removeWriter(PrintWriter writer)
{
    synchronized (listWriters)
    {
        listWriters.remove(writer);
    }
}
```

종료 요청 quit 처리 메서드
removeWriter 메서드를 통해
참여자 목록인 listWriters에서
종료자 제거

tokens[0]이 message인 경우

```
private void doMessage(String data)
{
    AES128의 경우 이 부분이 AES128.encrypt(message, ChatClientApp.SERVER_IP);
    String DecMessage = AES256.decrypt(data, ChatClientApp.SERVER_IP);
    broadcast(this.nickname + ":" + DecMessage);
}
```

```
private void broadcast(String data)
{
    synchronized (listWriters)
    {
        for(PrintWriter writer : listWriters)
        {
            writer.println(data);
            writer.flush();
        }
    }
}
```

AES로 복호화한 메시지를
Broadcast 메서드를 통해
모든 참여자에게 전송

②-6 ChatServerThread.java

tokens[0]이 file인 경우

```
private void doFile(String filename)
{
    //broadcast(filename); 잘 넘어옴
    //broadcast("DOFILE"); 잘 넘어옴
    try
    {
        OutputStream fos = new FileOutputStream(filename);
        InputStream is = socket.getInputStream();

        byte[] buffer = new byte[DEFAULT_BUFFER_SIZE];
        int readBytes = 0;

        while( (readBytes = is.read(buffer)) != -1 )
        {
            fos.write(buffer, 0, readBytes);
        }

        is.close();
        fos.close();
        //socket.shutdownInput();
        //이걸지운다is.close(); 말고 socket.shutdowninput();
    }
    catch(FileNotFoundException e)
    {
        System.out.println("ChatServerProcessThread : FileOutputStream File Not Found Exception");
    }
    catch(IOException e)
    {
        System.out.println("ChatServerProcessThread : InputStream IOException");
    }
}
```

ChatWindow의 sendFile 메서드에서 수신한 파일 이름과 파일 내용을 수신하여 처리하는 부분

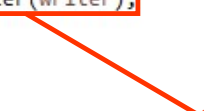
tokens[0]이 join인 경우

```
private void doJoin(String nickname, PrintWriter writer)
{
    this.nickname = nickname;

    String data = nickname + "님이 입장하였습니다.";
    broadcast(data);

    // writer pool에 저장
    addWriter(writer);
}
```

```
private void addWriter(PrintWriter writer)
{
    synchronized (listWriters)
    {
        listWriters.add(writer);
    }
}
```



누가 입장하였다고 broadcast한 후에
addWriter 메서드를 통해
입장자를 참여자 목록인 listWriters에 추가

②-7 AES.java

```
public class AES256
{
    public static String secretKey = ChatClientApp.SERVER_IP;
    public static String salt = "SALT";

    public static SecretKeySpec genSecretKey256(String secret)
    {
        AES128의 경우 genSecretKey128
        try
        {
            SecretKeyFactory factory = SecretKeyFactory.getInstance("PBKDF2WithHmacSHA256");
            KeySpec spec = new PBEKeySpec(secretKey.toCharArray(), salt.getBytes(), 65536, 256);
            SecretKey tmp = factory.generateSecret(spec);
            SecretKeySpec secretKey = new SecretKeySpec(tmp.getEncoded(), "AES");
            AES128의 경우 128

            return secretKey;
        }
        catch (Exception e)
        {
            System.out.println("Error while genSecretKey : " + e.toString());
        }
        return null;
    }
}
```

서버IP주소와 솔트값을 이용해
비밀키 생성하여 암호/복호화에 사용

②-7 AES.java

```
public static String encrypt(String strToEncrypt, String secret)
{
    try
    {
        byte[] iv = { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 };
        IvParameterSpec ivspec = new IvParameterSpec(iv);

        SecretKeySpec secretKey = genSecretKey256(secret);
        AES의 경우 genSecretKey128
        Cipher cipher = Cipher.getInstance("AES/OFB/PKCS5Padding");
        cipher.init(Cipher.ENCRYPT_MODE, secretKey, ivspec);
        return Base64.getEncoder().encodeToString(cipher.doFinal(strToEncrypt.getBytes("UTF-8")));
    }
    catch (Exception e)
    {
        System.out.println("Error while encrypting: " + e.toString());
    }
    return null;
}

public static String decrypt(String strToDecrypt, String secret)
{
    try
    {
        byte[] iv = { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 };
        IvParameterSpec ivspec = new IvParameterSpec(iv);

        SecretKeySpec secretKey = genSecretKey256(secret);
        AES의 경우 genSecretKey128
        Cipher cipher = Cipher.getInstance("AES/OFB/PKCS5PADDING");
        cipher.init(Cipher.DECRYPT_MODE, secretKey, ivspec);
        return new String(cipher.doFinal(Base64.getDecoder().decode(strToDecrypt)));
    }
    catch (Exception e)
    {
        System.out.println("Error while decrypting: " + e.toString());
    }
    return null;
}
```

PKCS5 패딩을 사용하여
AES/OFB모드에서
암/복호화 동작

②-8 구현 결과

ChatServer [Java Application]
연결 기다림 - 192.168.37.1:5000

ChatClientApp [Java Application]
ENTER YOUR NICKNAME : SAMSUNG
YOU HAVE ENTERED CHATTING ROOM!!

ChatClientApp [Java Application]
ENTER YOUR NICKNAME : LG
YOU HAVE ENTERED CHATTING ROOM!!

SAMSUNG LG님이 입장하셨습니다.
SAMSUNG:HI LG-
SAMSUNG:안녕하세요 LG-
LG:HI SAMSUNG-
LG:안녕하세요 삼성~

Send Attach

LG SAMSUNG:HI LG-
SAMSUNG:안녕하세요 LG-
LG:HI SAMSUNG-
LG:안녕하세요 삼성~

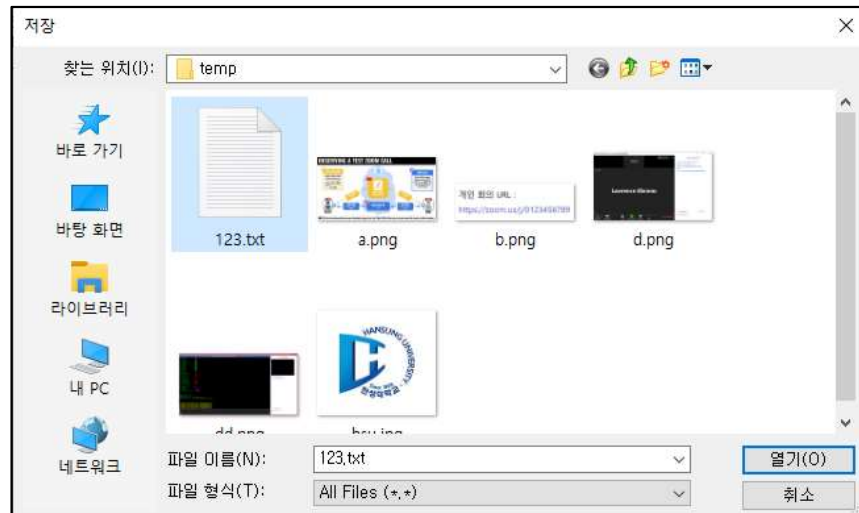
Send Attach

① 서버 OPEN

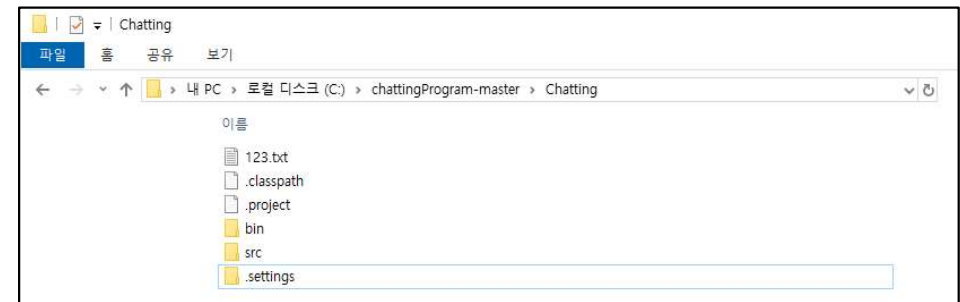
② 클라이언트 접속

③ 채팅 진행

②-8 구현 결과



④ Attach 버튼 클릭 시
FileDialog 실행



⑤ 파일 전송 완료(123.txt)

②-8 구현 결과

```
ChatServer [Java Application]
연결 기다림 - 192.168.37.1:5000
SAMSUNG님이 채팅방을 나갔습니다.
Size of File : 0
Delete success.
Size of File : 0
Delete success.
LG님이 채팅방을 나갔습니다.
Size of File : 0
Delete success.
```

클라이언트 접속 종료 후
서버 로그 삭제

```
ChatClientApp [Java Application]
ENTER YOUR NICKNAME : LG
YOU HAVE ENTERED CHATTING ROOM!!
Size of File : 0
Delete success.
```

```
ChatClientApp [Java Application]
ENTER YOUR NICKNAME : SAMSUNG
YOU HAVE ENTERED CHATTING ROOM!!
Size of File : 0
Delete Failed.
```

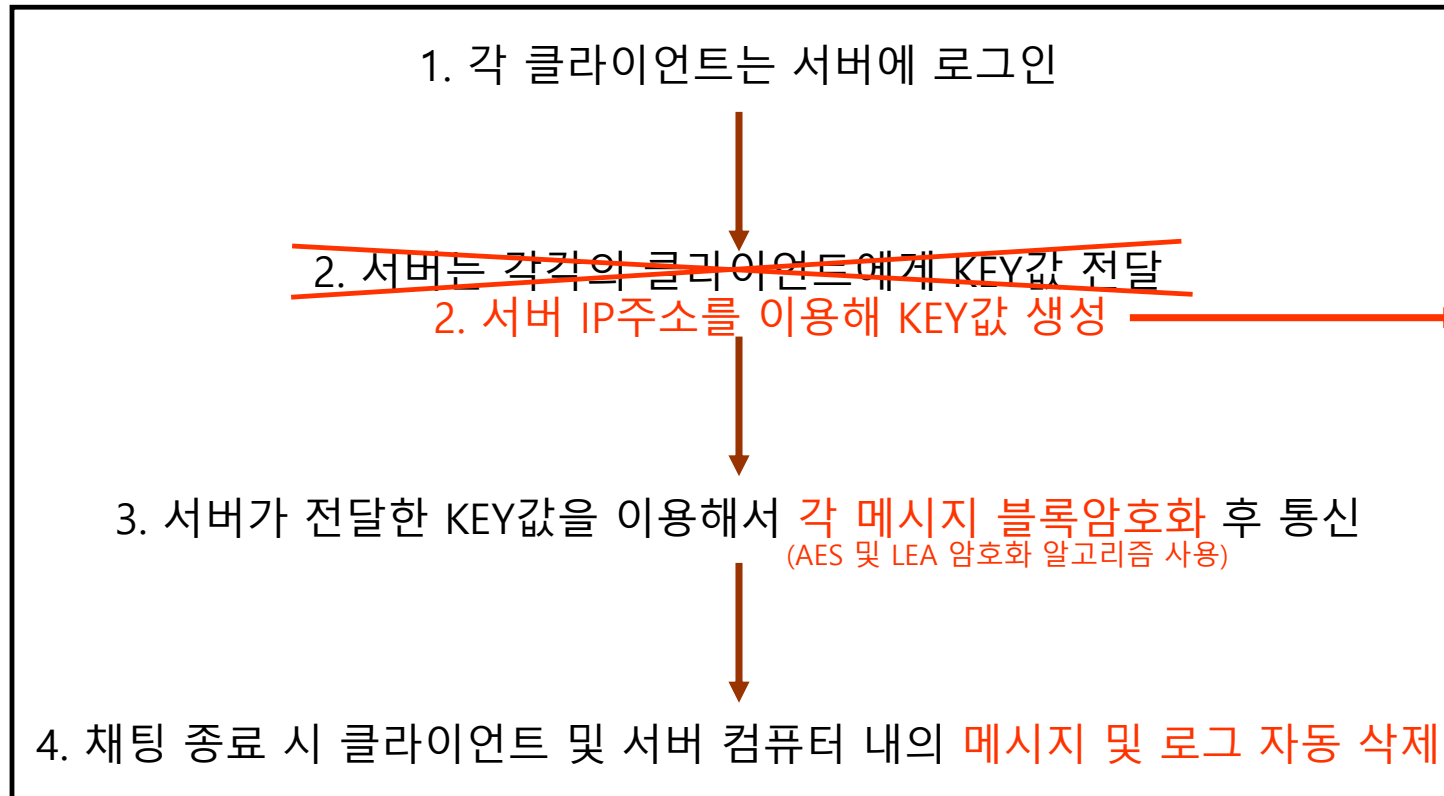
클라이언트 접속 종료 후
클라이언트 로그 삭제

자세한 구현사항은 시연동영상으로 첨부하였습니다.

③ 개발 과정에서의 문제점 해결

③-1 KEY값 생성

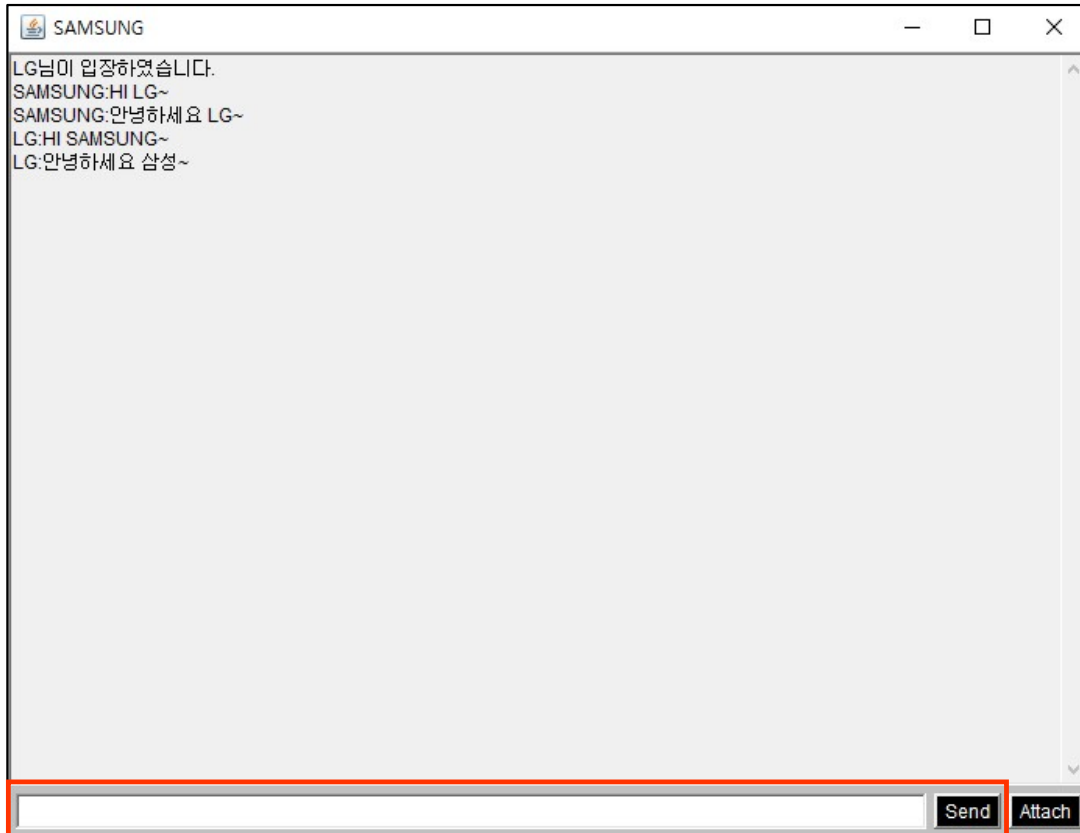
프로그램 동작 방법



변경 이유?

서버 KEY값 전달 시
KEY값 노출 위험 존재
(맨 처음에 무조건
최소 한 번 송수신
되기 때문)

③-2 채팅 송신 오류 및 해결



아무것도 입력하지 않고 Send할 때
채팅이 진행이 되지 않고 끝나는 오류

```
@Override
public void run()
{
    while(true)
    {
        try
        {
            BufferedReader buffereedReader =
                new BufferedReader(new InputStreamReader(socket.getInputStream(), StandardCharsets.UTF_8));

            PrintWriter printWriter =
                new PrintWriter(new OutputStreamWriter(socket.getOutputStream(), StandardCharsets.UTF_8));

            String request = buffereedReader.readLine();

            if(request == null)
            {
                consoleLog("클라이언트로부터 연결 끊김");
                doQuit(printWriter);
                break;
            }

            String[] tokens = request.split(":");
            if("join".equals(tokens[0]))
            {
                doJoin(tokens[1], printWriter);
            }
            else if("message".equals(tokens[0]))
            {
                doMessage(tokens[1]);
            }
            else if("quit".equals(tokens[0]))
            {
                doQuit(printWriter);
            }
            else if("file".equals(tokens[0]))
            {
                doFile(tokens[1]);
            }
        }
    }
}
```

ChatServerThread.java의 bufferedReader에
아무것도 입력이 되지 않아서 if문에 걸리지 않고
그대로 스레드가 종료되어서 발생한 오류
run() 부분을 무한루프 시켜주어서 해결

③-3 서버 접속 불가 오류 및 해결

ChatClientApp [Java Application]
ENTER YOUR NICKNAME : SAMSUNG

서버가 열리지 않았을 시에
접속이 불가능해도 계속 접속 시도

```
Socket socket = new Socket();
try
{
    socket.connect(new InetSocketAddress(SERVER_IP, SERVER_PORT));

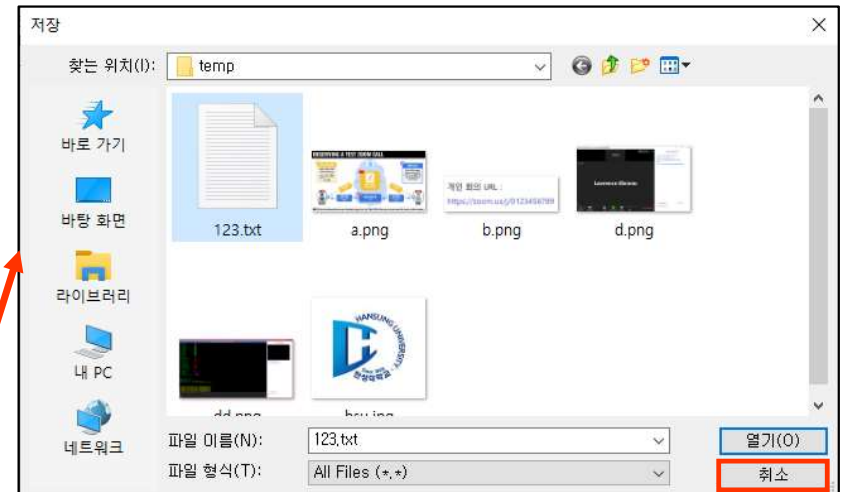
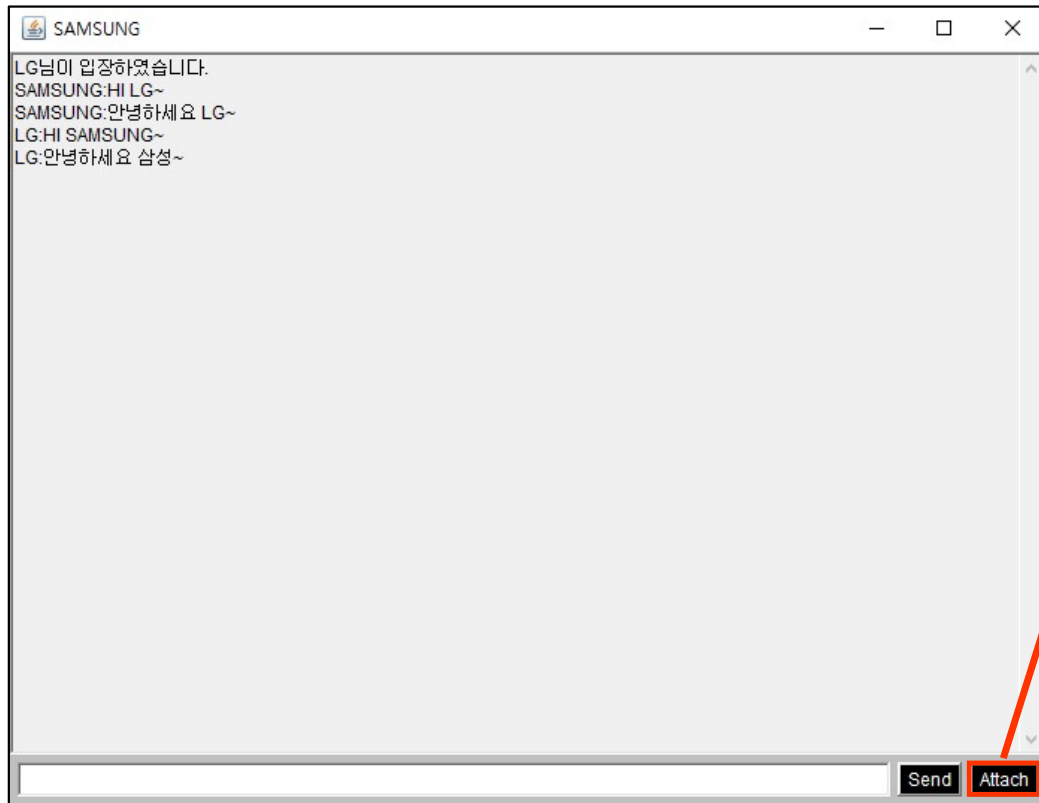
    consoleLog("YOU HAVE ENTERED CHATTING ROOM!!");
    //여기서 ChatWindow에 name과 socket을 가지고 넣어간다.
    new ChatWindow(name, socket).show();
    PrintWriter pw =
        new PrintWriter(new OutputStreamWriter(socket.getOutputStream(), StandardCharsets.UTF_8), true);
    String request = "join:" + name + "\r\n";
    pw.println(request);
}
catch (ConnectException e)
{
    System.out.println("서버와의 연결이 거부되었습니다.");
    System.out.println("시스템을 종료합니다.");
    System.exit(0);
}
catch (IOException e)
{
    e.printStackTrace();
    System.out.println("IOException이 발생하였습니다.");
    System.out.println("시스템을 종료합니다.");
    System.exit(0);
}
```

ChatClient.java에서
ConnectException 처리하여
서버가 열리지 않았을 때
프로그램을 종료하도록 처리

ChatClientApp [Java Application]
ENTER YOUR NICKNAME : SAMSUNG
서버와의 연결이 거부되었습니다.
시스템을 종료합니다.

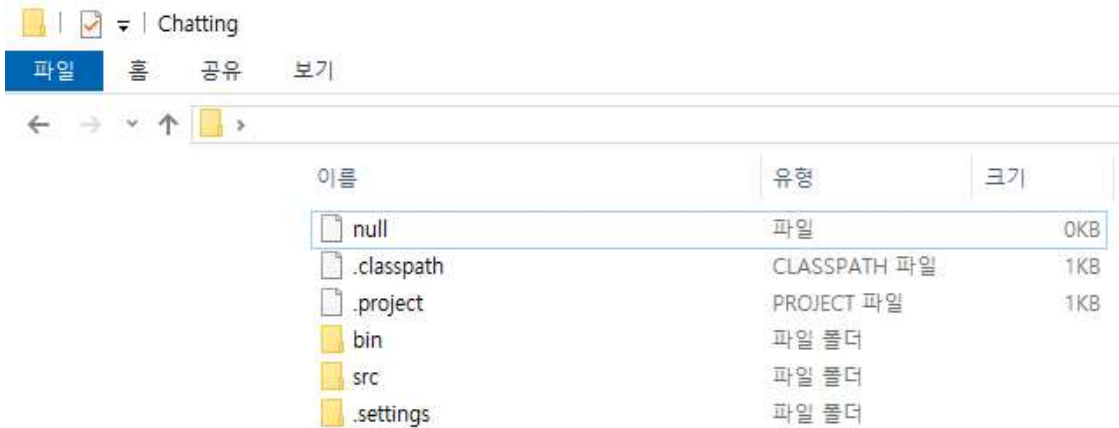
결과창

③-4 파일 첨부 오류 및 해결



취소 버튼 클릭 시에
채팅이 송수신되지 않던 오류

③-4 파일 첨부 오류 및 해결



null 파일 수신 후 채팅 종료됨
취소했을 시의 조건문이 없어서 발생한 오류

```
String dir = dialog.getDirectory();
String filename = dialog.getFile();
String FilePath = dir + filename;
//System.out.println(FilePath);

File AttachFile = new File(FilePath);
if(!AttachFile.exists())
{
    System.out.println("File Doesn't Exist!!");
    return;
}
```

파일을 선택하지 않아서
File이 열리지 않는 경우의 조건문을
삽입하여 아무 동작도 하지 않도록
설정하여 오류 해결

④ 마무리

④-1 정리

AES-128/256 or LEA-128/256으로
송수신 내용 블록암호화 구현 성공 → 메신저 내용 도청 방지

채팅 종료 후
채팅 기록, 로그 자동 삭제 → 흔적 제거를 통해
해킹 시 개인정보 노출 방지

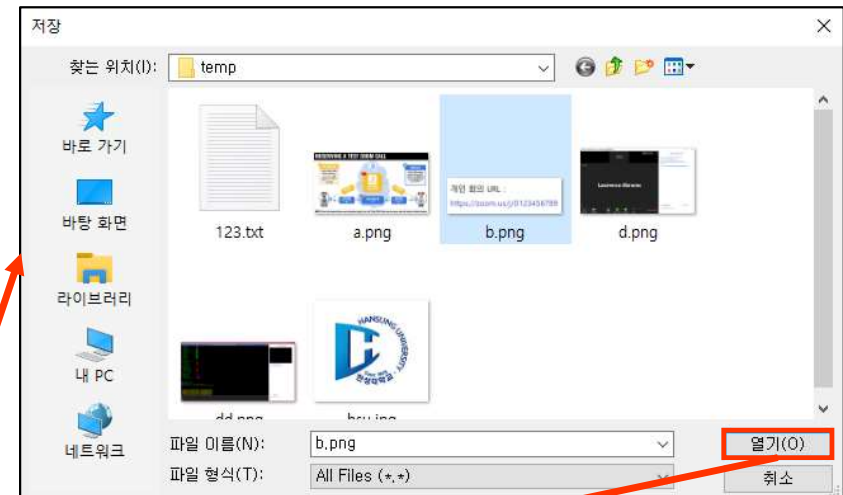
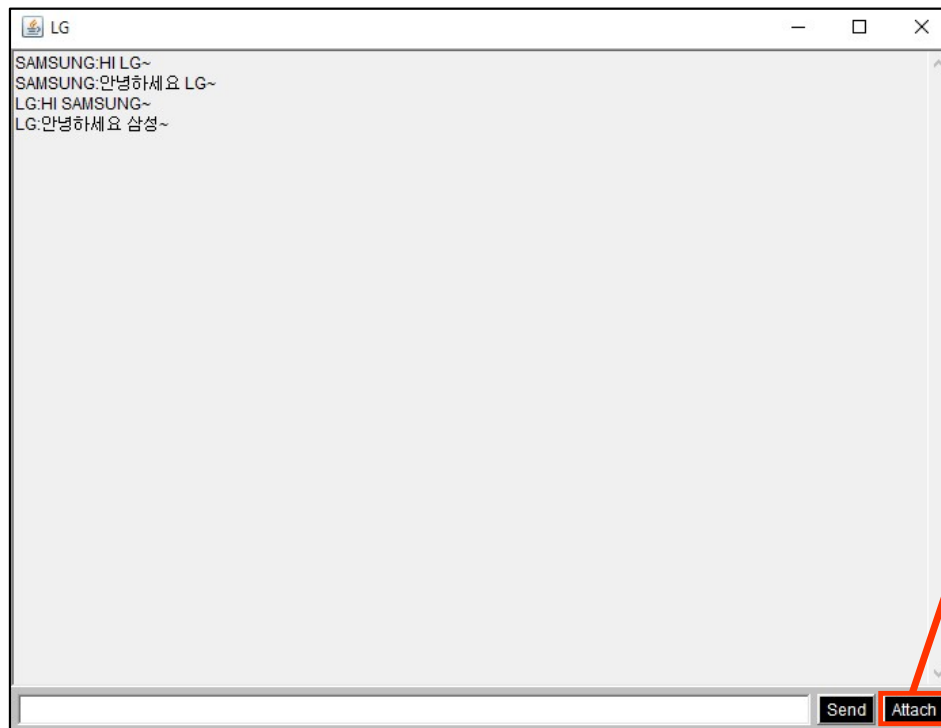
메신저 해킹, 유출 시 개인정보 노출 방지 가능!!

하지만 복잡한 비밀번호 설정, 2차 비밀번호 설정 등
사용하기 번거롭지만 강력한 보안 절차 사용이 가장 중요!!

또한 메신저 해킹 시의 개인정보 노출 방지를 위해
필요 없는 채팅은 삭제하는 등 정리하는 습관도 중요!!

④-2 추후 보완 사항

파일 전송 후 채팅 진행 시 오류 발생
ChatWindow.java의 sendFile() 메서드



```
ChatClientApp [Java Application]
ENTER YOUR NICKNAME : LG
YOU HAVE ENTERED CHATTING ROOM!!
Send File execute.
File Transfer completed.
java.net.SocketException: Socket closed
    at java.base/sun.nio.ch.NioSocketImpl.endRead(NioSocketImpl.java:249)
    at java.base/sun.nio.ch.NioSocketImpl.implRead(NioSocketImpl.java:328)
    at java.base/sun.nio.ch.NioSocketImpl.read(NioSocketImpl.java:351)
    at java.base/sun.nio.ch.NioSocketImpl$1.read(NioSocketImpl.java:802)
    at java.base/java.net.Socket$SocketInputStream.read(Socket.java:937)
    at java.base/sun.nio.cs.StreamDecoder.readBytes(StreamDecoder.java:297)
    at java.base/sun.nio.cs.StreamDecoder.implRead(StreamDecoder.java:339)
    at java.base/sun.nio.cs.StreamDecoder.read(StreamDecoder.java:188)
    at java.base/java.io.InputStreamReader.read(InputStreamReader.java:185)
    at java.base/java.io.BufferedReader.fill(BufferedReader.java:161)
    at java.base/java.io.BufferedReader.readLine(BufferedReader.java:326)
    at java.base/java.io.BufferedReader.readLine(BufferedReader.java:392)
    at ChatWindow$ChatClientReceiveThread.run(ChatWindow.java:285)
```


④-2 추후 보완 사항

파일 전송 후 채팅 진행 시 오류 발생
ChatWindow.java의 sendFile() 메서드

```
byte[] buffer = new byte[DEFAULT_BUFFER_SIZE];
int readBytes = 0;
InputStream fis;
OutputStream os;
try
{
    fis = new FileInputStream(FilePath);
    os = socket.getOutputStream();
    while((readBytes = fis.read(buffer)) > 0)
    {
        os.write(buffer, 0, readBytes);
    }
    os.flush();
    os.close();
    fis.close();
}
catch(FileNotFoundException e)
{
    System.out.println("ChatWindow : FileNotFoundException");
    e.printStackTrace();
}
catch(IOException e)
{
    System.out.println("ChatWindow : IOException");
    e.printStackTrace();
}
System.out.println("File Transfer completed.");
```

이 부분에서 오류가 난 것으로 추정
Future Work에 포함하여 추후 해결 예정

해당 부분 소스 코드
ChatWindow.java 의 sendFile() 메서드

④-2 추후 보완 사항

블록암호화 알고리즘(AES-128,256 or LEA-128,256)을 적용하여 파일 전송 시 오류 발생
ChatWindow.java의 sendFile() 메서드

```
try
{
    InputStream fis = new FileInputStream(FilePath);
    OutputStream os = socket.getOutputStream();
    while((readBytes = fis.read(buffer)) > 0)
    {
        os.write(buffer, 0, readBytes);
    }

    fis.close();
    os.flush();
    os.close();
    //os.close();
    //socket.shutdownOutput();
}
catch(FileNotFoundException e)
{
    System.out.println("ChatWindow : FileNotFoundException");
    e.printStackTrace();
}
catch(IOException e)
{
    System.out.println("ChatWindow : IOException");
    e.printStackTrace();
}

System.out.println("File Transfer completed.");
}
```

InputStream, OutputStream, byte[] buffer를
암/복호화해야 한다고 추정
Future Work에 포함하여 추후 해결 예정

④-3 Future Work

Future Work	
주차	활동
1주차	파일 전송 후 채팅 진행 불가능 오류 수정
2주차 ~ 3주차	파일 전송 블록암호화 알고리즘 (AES-128, 256 or LEA-128, 256) 적용
3주차 ~ 4주차	취약점 테스트 설계 및 진행
4주차 ~ 7주차	스마트폰용 어플리케이션 개발
~	교내외 사이버보안 관련 공모전 참가 예정

버퍼 오버플로우 공격 등을 통해
시스템 공격 가능성 테스트,
WireShark를 이용한
도청 테스트,
서버 컴퓨터 침입 테스트 예정

④-4 개발 일정 및 역할 분담

진행 사항		역할 분담	
주차	활동	이름	역할
3주차	제안서 제출	이광헌	프로젝트 진행 및 프로그램 개발 보조
4주차~8주차	프로그램 개발(텍스트 채팅)	배재현	발표 및 발표 자료 디자인
8주차~9주차	중간 발표 자료 제출	나관우	프로그램 개발 총괄
9주차~11주차	파일 첨부 기능 개발		
12주차~14주차	메시지 블록암호화 적용		
15주차	각종 오류 수정 및 인터페이스 수정		

④-5 지원금 집행 내역

N/A

④-6 참고 문헌

주진모 카카오톡 해킹 관련 기사
해킹 피해 주진모, '지라시'에 법적대응 - 뉴스컬처
<http://nc.asiae.co.kr/view.htm?idxno=2020011015030506388>

피해사례 자료
구글 해킹 사건의 발단은 '직원 메신저'였다! - 보안뉴스
<https://www.boannews.com/media/view.asp?idx=20540&page=3008&kind=3>

피해사례 자료
카톡보다 안전하다는 메신저 '텔레그램'도 해킹 당했다. - 조선일보
https://news.chosun.com/site/data/html_dir/2018/10/01/2018100100209.html

LEA 관련 자료
LEA - 해시넷
<http://wiki.hash.kr/index.php/LEA>

LEA 관련 자료
LEA - 나무위키
<https://namu.wiki/w/LEA>

Java 기초 학습
자바 시작 - TCP SCHOOL
<http://tcpschool.com/java/intro>

Java 소켓 프로그래밍(멀티스레드)
멀티스레드를 이용한 에코서버 - 개발하는 조세이돈
<https://devshock.tistory.com/48>

Java 소켓 프로그래밍(멀티스레드)
Java TCP 소켓 프로그래밍 예제 - 정아마추어 코딩블로그
<https://jeong-pro.tistory.com/134>

AES 암호/복호화 예제
Java AES Example - HowDoInJava
<https://howtodoinjava.com/security/aes-256-encryption-decryption/>

LEA 알고리즘 패키지
블록암호 LEA - KISA 암호이용활성화
<https://seed.kisa.or.kr/kisa/Board/20/detailView.do>

감사합니다!!