

# 시스템 및 네트워크보안

- 3주차 과제 : 게이트 웨이를 거치는 서버-클라이언트 통신 구현 -

201002472 이우람

201102518 최인동

## 1. 프로그램 소개

이 프로그램은 client와 server가 서로 메시지를 전달하는 채팅 프로그램입니다. client에서 메시지를 입력하고 gateway를 통해 server로 전달합니다. 이후 server에서는 client에서 메시지를 받아 출력한 후에 server에서도 gateway를 통해 client로 메시지를 전달합니다.

## 2. 핵심코드 설명

```
send_message = new Message(server_addr,server_port,
    client_addr, client_port,"data",null,client_publicKey);
oos.reset();
oos.writeObject(send_message);
// client public key를 server에게 전송

Message rcv_message = (Message)ois.readObject();
Key server_publicKey = rcv_message.public_key;
// server에서 server public key를 받음

System.out.print("Send Data (>\"exit\" => disconnect)");
/*send data*/
while(true){
    System.out.print(">");
    String str = scan.nextLine();
    if(str.equals("exit")) break;

    byte[] cipherText = testRSA_encrypt(server_publicKey,str); // 입력한 문자를 server의 public key로 암호화
    //System.out.println("rsa">"+byteArrayToHex(cipherText)); // 암호화된 문자 확인
    send_message = new Message(server_addr,server_port,
        client_addr, client_port,"data",byteArrayToHex(cipherText), null); // 암호화된 문자를 전송
    oos.reset();
    oos.writeObject(send_message);

    rcv_message = (Message)ois.readObject();
    //System.out.println("rsa">"+rcv_message.msg); // 암호화된 문자 확인
    System.out.println("server">"+
        new String(testRSA_decrypt(client_privateKey, hexToByteArray(rcv_message.msg)))); // public private key로 복호화
}
```

< ClientMain.java의 일부분 >

위 부분에서는 RSA함수를 통해 생성한 Client 공개키를 Server에게 전송하고, 보내는 메시지를 암호화하기 위해 Server의 공개키를 받습니다. Server로 보낼 메시지를 입력받고 Server의 공개키로 암호화하여 전송합니다. 그 이후에는 Server로부터 오는 메시지를 기다리고, 해당 메시지를 Client의 개인키로 복호화 하여 출력합니다.

```
Message rcv_message = (Message)ois.readObject();
Key client_publicKey = rcv_message.public_key;
// client에서 client public key를 받음

Message send_message = new Message(rcv_message.src_addr, rcv_message.src_port,
    socket.getLocalAddress().toString(),port,"data",null,server_publicKey);
oos.reset();
oos.writeObject(send_message);
// server public Key를 client에게 전송

while(true){
    rcv_message = (Message)ois.readObject();
    //System.out.println("rsa">"+rcv_message.msg); // 암호화된 문자 확인
    System.out.println("client">"+
        new String(testRSA_decrypt(server_privateKey, hexToByteArray(rcv_message.msg)))); // server private key로 복호화

    System.out.print(">");
    String str = scan.nextLine();
    // 보낼 문자 입력

    byte[] cipherText = testRSA_encrypt(client_publicKey,str); // 입력한 문자를 server의 public key로 암호화
    //System.out.println("rsa">"+byteArrayToHex(cipherText)); // 암호화된 문자 확인
    send_message = new Message(rcv_message.src_addr, rcv_message.src_port,
        socket.getLocalAddress().toString(),port,"data",byteArrayToHex(cipherText), null); // 암호화된 문자 전송
    oos.reset();
    oos.writeObject(send_message);
}
```

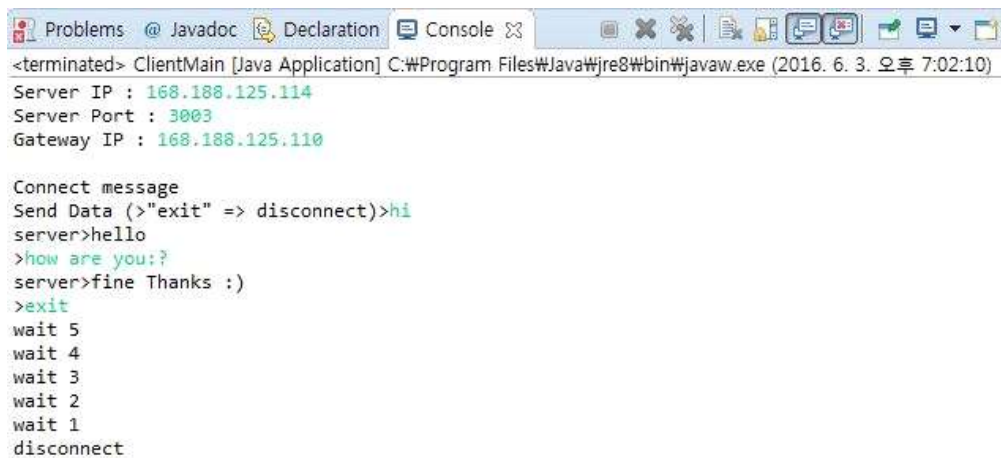
< ServerMain.java의 일부분 >

위 코드 부분에서는 먼저 Client로부터 Client의 공개키를 받고, Server의 공개키를 전송합니다. 그리고 받은 메시지를 Server의 개인키로 복호화하여 출력해줍니다. 그이후에 입력한 메시지를 Client의 공개키로 암호화하여 메시지를 보내줍니다.

### 3. 적용한 보안 알고리즘

채팅 메시지를 순서대로 전달하는 과정에서는 공개키-개인키 암호화 방식(RSA 방식)을 적용하였습니다. 이러한 방식을 적용한 이유는 client와 server가 메시지를 주고받을 때 마다 gateway를 통해 패킷을 전송하는데, 이 과정에서 불미스러운 사고로 패킷이 노출된다 하더라도 메시지는 이미 서로의 공개키로 암호화 되어 있습니다. 그 메시지들은 각자가 가지고 있는 개인키가 없으면 복호화가 불가능하므로 강력한 보안 대책이 될 수 있습니다.

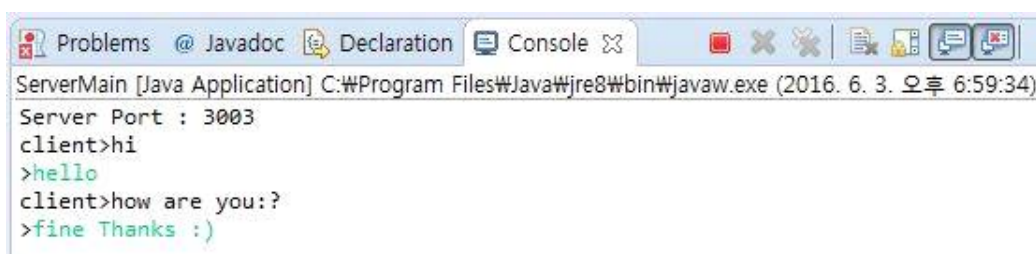
### 4. 결과 캡처 화면



```
<terminated> ClientMain [Java Application] C:\Program Files\Java\jre8\bin\javaw.exe (2016. 6. 3. 오후 7:02:10)
Server IP : 168.188.125.114
Server Port : 3003
Gateway IP : 168.188.125.110

Connect message
Send Data (>"exit" => disconnect)>hi
server>hello
>how are you:?
server>fine Thanks :)
>exit
wait 5
wait 4
wait 3
wait 2
wait 1
disconnect
```

< Client console >



```
ServerMain [Java Application] C:\Program Files\Java\jre8\bin\javaw.exe (2016. 6. 3. 오후 6:59:34)
Server Port : 3003
client>hi
>hello
client>how are you:?
>fine Thanks :)
```

< Server console >

Filter:	ip.addr == 168.188.125.110 && ip.addr == 168.188.125.114	Expression...	Clear	Apply
Source	Destination	Protocol	Length	Info
168.188.125.110	168.188.125.114	IPA	590	RSL [Malformed Packet]
168.188.125.110	168.188.125.114	IPA	238	unknown 0x38 [Malformed Packet]
168.188.125.114	168.188.125.110	TCP	60	3003->49457 [ACK] Seq=743 Ack=146
168.188.125.114	168.188.125.110	IPA	60	[Malformed Packet]
168.188.125.114	168.188.125.110	IPA	590	RSL [Malformed Packet]
168.188.125.114	168.188.125.110	IPA	239	unknown 0x39 [Malformed Packet]
168.188.125.110	168.188.125.114	TCP	54	49457->3003 [ACK] Seq=1463 Ack=14
168.188.125.110	168.188.125.114	TCP	55	3000->63326 [PSH, ACK] Seq=743 Ac
168.188.125.110	168.188.125.114	TCP	590	3000->63326 [PSH, ACK] Seq=744 Ac
168.188.125.110	168.188.125.114	TCP	239	3000->63326 [PSH, ACK] Seq=1280 A
168.188.125.114	168.188.125.110	TCP	60	63326->3000 [ACK] Seq=1673 Ack=14
168.188.125.114	168.188.125.110	TCP	60	63326->3000 [PSH, ACK] Seq=1673 A
168.188.125.114	168.188.125.110	TCP	590	63326->3000 [PSH, ACK] Seq=1674 A
168.188.125.114	168.188.125.110	TCP	238	63326->3000 [PSH, ACK] Seq=2210 A
168.188.125.110	168.188.125.114	TCP	54	3000->63326 [ACK] Seq=1465 Ack=23
168.188.125.110	168.188.125.114	IPA	55	[Malformed Packet]
168.188.125.110	168.188.125.114	IPA	590	RSL [Malformed Packet]
168.188.125.110	168.188.125.114	IPA	238	unknown 0x65 [Malformed Packet]
168.188.125.114	168.188.125.110	TCP	60	3003->49457 [ACK] Seq=1465 Ack=21
168.188.125.114	168.188.125.110	IPA	60	[Malformed Packet]
168.188.125.114	168.188.125.110	IPA	590	RSL [Malformed Packet]
168.188.125.114	168.188.125.110	IPA	239	unknown 0x66 [Malformed Packet]

< 패킷 캡처 일부분 >

00e0	2e 31 32 35 2e 31 31 34	74 02 00 38 36 63 63 30	.125.114 t..86cc0
00f0	37 31 35 34 64 35 35 37	35 39 31 32 62 31 61 34	7154d557 5912b1a4
0100	32 66 62 32 66 39 61 39	35 62 31 37 37 61 33 31	2fb2f9a9 5b177a31
0110	63 64 38 36 65 64 31 66	36 36 61 61 30 33 31 38	cd86ed1f 66aa0318
0120	32 33 32 35 37 64 34 36	61 39 35 35 65 65 35 64	23257d46 a955ee5d
0130	62 32 32 36 39 66 35 36	32 64 30 65 32 65 33 38	b2269f56 2d0e2e38
0140	36 31 63 63 36 30 65 38	62 36 64 32 65 62 65 63	61cc60e8 b6d2ebec
0150	63 64 66 39 61 30 35 30	30 36 63 65 35 33 31 32	cdf9a050 06ce5312
0160	66 34 35 66 65 38 61 63	36 62 30 30 37 63 31 66	f45fe8ac 6b007c1f
0170	64 62 64 38 63 63 39 65	37 33 31 35 61 63 31 62	dbd8cc9e 7315ac1b

< 선택한 패킷의 내용 - gateway log에 나온 패킷과 일치>

```

@ Javadoc Declaration Console
GatewayMain [Java Application] C:\Program Files\Java\jre1.8.0_73\bin\javaw.exe (2016. 6. 3. 오후 7:03:41)
Client->Gateway send message [null]
Server->Gateway send message [null]
Client->Gateway send message [86cc07154d5575912b1a42fb2f9a95b177a31cd86ed1f6]
Gateway->Server send message [86cc07154d5575912b1a42fb2f9a95b177a31cd86ed1f6]
Server->Gateway send message [8df283adeea43982ce4797f2e97883ef4e70116781abf5]
Gateway->Server send message [8df283adeea43982ce4797f2e97883ef4e70116781abf5]
Client->Gateway send message [5a8a48ff9ddb961667eedcdcfeb85e54ed5d72f7357cc8]
Gateway->Server send message [5a8a48ff9ddb961667eedcdcfeb85e54ed5d72f7357cc8]
Server->Gateway send message [39f9e180b4f45e399d4efd1f1f6664e7ab87672e429425]
Gateway->Server send message [39f9e180b4f45e399d4efd1f1f6664e7ab87672e429425]
Client->Gateway send message [null]
thread terminated

```

< Gateway console >