**네트워크 보안**

3주차 Gateway를 거치는 통신

|  |  |
| --- | --- |
| 학번 | 이름 |
| 201002424 | 박지현 |
| 201102435 | 박민수 |
| 201102479 | 이겨례 |

1. **프로그램 소개**

* **프로그램 목적**

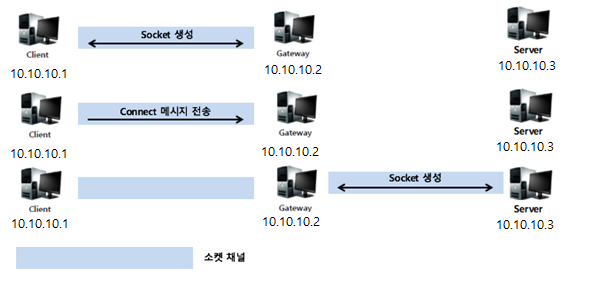
클라이언트 – 게이트웨이 – 서버의 통신 프로그램을 구현하고 클라이언트와 서버 사이에 보안을 적용하여 종단간의 암호화 통신을 구현한다.

* **구현내용**

Client와 Server 사이에 RSA 보안을 적용하여 통신 구간 상에서 패킷을 캡처해도 메시지의 내용이 노출되지 않는 채팅 프로그램을 구현한다. Gateway에서 넘겨주는 데이터의 내용은 로그로 출력하여 확인한다.

* **동작 과정**

1) 연결(소켓 생성)



Client와 Gateway사이에 Socket이 생성되면 Client는 connect 메시지를 Gateway에게 전송한다. 그러면 Gateway와 Server사이에 Socket이 생성되고 연결이 완료된다.

2) 데이터 전송



* Client -> Server

: Server는 공개 키와 비밀 키를 생성해서 Client에게 공개 키를 전송해준다. Client는 자신이 받은 Server의 공개키로 메세지를 암호화해서 Gateway로 암호문을 전송하면 Gateway는 Server에게 암호문을 전달한다.

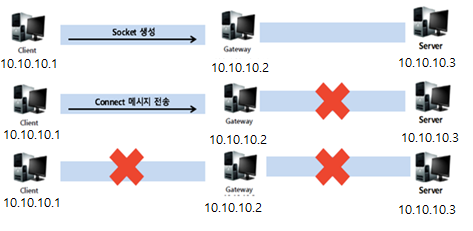
Server는 Client로부터 받은 메세지를 자신의 비밀키로 복호화하여 평문을 얻는다.

* Server -> Client

: Client는 공개 키와 비밀 키를 생성해서 Server에게 공개 키를 전송한다. Server는 자신이 받은 Client의 공개키로 메시지를 암호화해서 Gateway로 암호문을 전송하면 Gateway는 Client에게 암호문을 전달한다.

Client는 Server로부터 받은 메시지를 자신의 비밀키로 복호화하여 평문을 얻는다.

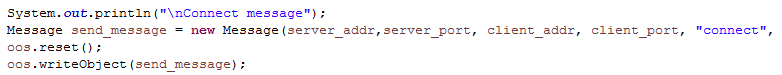
3) 연결 해제



Gateway는 Server와의 연결을 먼저 해제한 후, Client와의 연결을 해제한다.

1. **핵심기능의 소스코드 설명**

* **Client**



* 연결을 위해 Gateway에게 connect메시지를 전송.



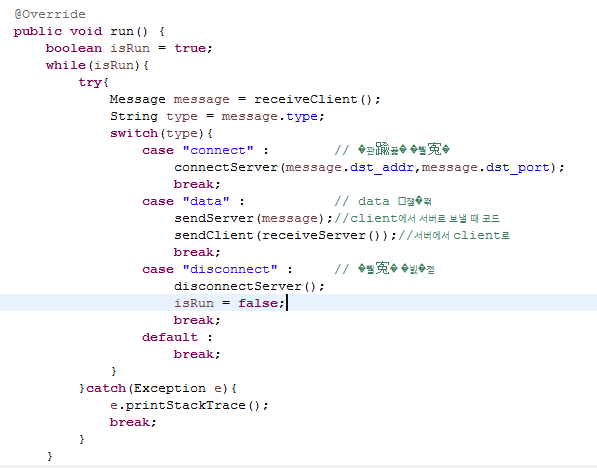
* Server에게 start메시지를 전송하면 채팅이 시작.



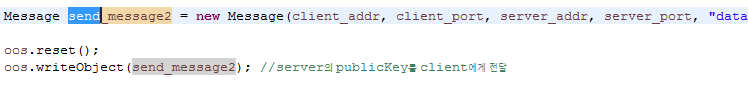
* Server에게 받은 메시지에서 publicKey를 server\_publicKey라는 변수에 저장.



* Client 먼저 채팅창에 메시지를 입력할 수 있다. 만일 exit을 입력하면 disconnect된다.
* 입력한 메시지는 처음에 받았던 Server의 publicKey로 암호화해서 암호문을 Gateway에 전송하면 Gateway는 Server에게 암호문을 전달한다.
* 만일 Server가 Client에게 메시지를 보내는 경우에는 자신의 비밀키로 복호화해서 채팅창에 표시하게 된다.
* **Gateway**
* ConnectThread.java



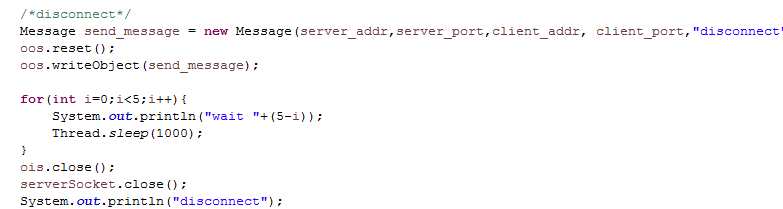
* Gateway에게 넘어온 메세지의 type이 connect인 경우 connectServer함수를 호출하여 연결을 완료시킨다.
* 메시지의 type이 data인 경우, client에게서 받은 메시지는 server에게 전달하고 server에게서 받은 메시지는 client에게 전달한다.
* Disconnect인 경우에는 server의 소켓을 닫아서 연결을 해제한다.
* **Server**



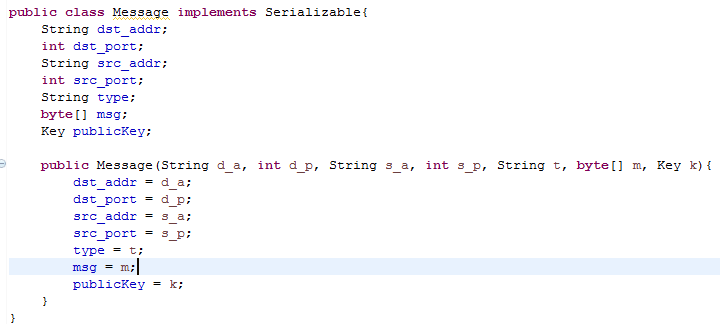
* Server는 자신의 public key를 client에게 전달.



* start라는 메시지가 뜨면서 채팅이 시작. client에게서 받은 메시지를 자신의 비밀키로 복호화해서 화면에 출력해주고, server가 입력할 차례가 되면 앞에서 받은 client의 공개키로 메시지를 암호화해서 보낸다.



* exit메시지가 입력되면 5초간 기다린 후에 소켓을 해제.
* **공통 Message.java**



* Message의 구조는 위와 같다.

1. **적용한 보안 알고리즘에 대한 설명**

* **사용 알고리즘**

: RSA 알고리즘

* **알고리즘 선택 이유**

: 공개키 알고리즘은 암호화 키와 복호화 키가 다르며 키 관리와 키 분배가 용이하다. 공개키 암호화는 사용자와 키를 공유하지 않고 암호화를 통한 안전한 통신을 한다는 장점이 있기 때문에 대표적인 공개키 암호화 알고리즘인 RSA방식을 선택하였다.

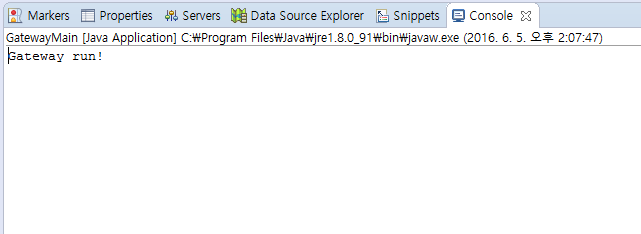
* **적용한 보안 알고리즘에 대한 설명**

: 송신자는 수신자의 공개 키를 받아 데이터를 암호화하여 네트워크를 통해 원격지에 전송한다. 수신자는 공개 키를 암호화된 데이터를 자신의 개인키로 데이터를 복호화 하여 복호화된 데이터를 얻는다..

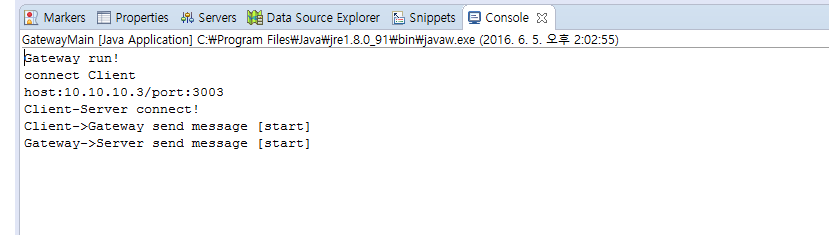
1. **결과화면**

* **연결**

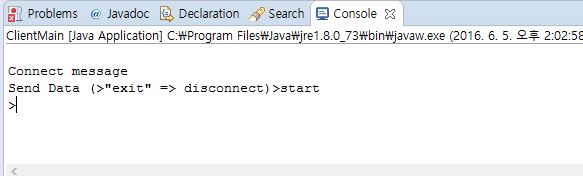
1) Gateway 초기



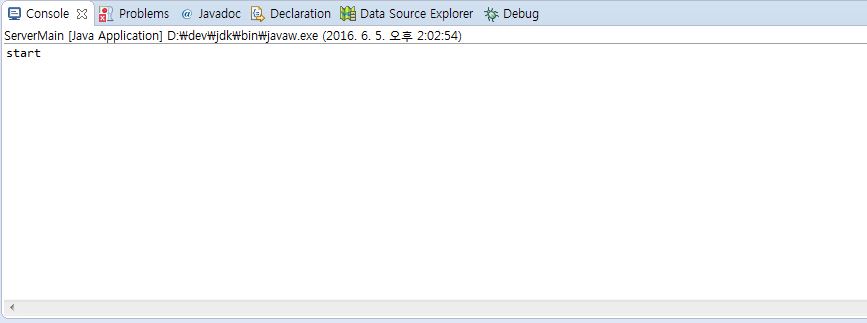
2) Gateway 연결되었을 때



3) client 첫 연결

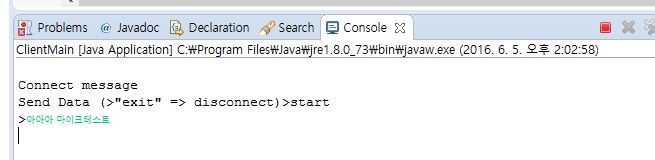


4) sever 첫연결

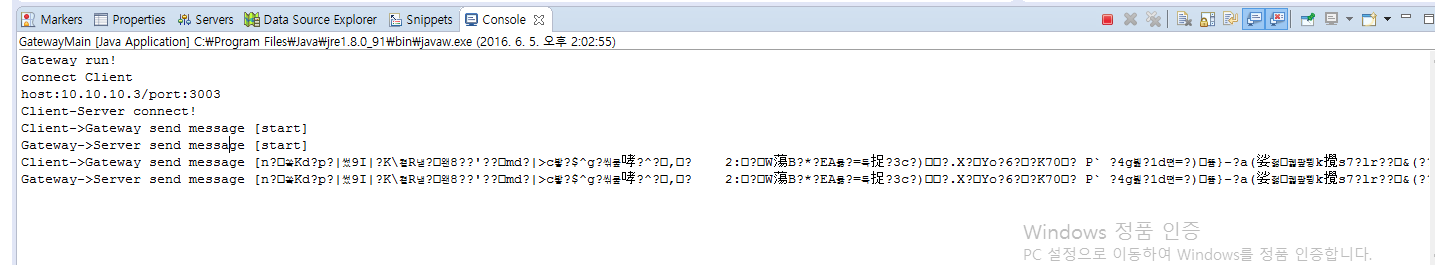


* **데이터전송1 Client → Server**

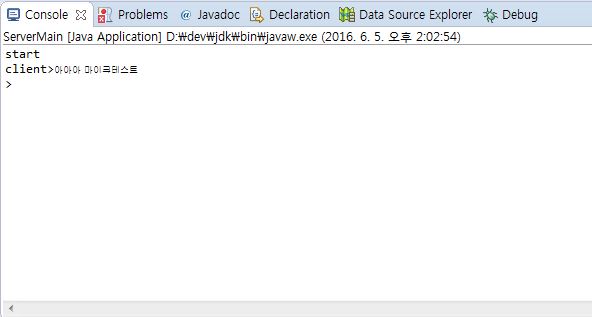
1) Client



2) Gateway

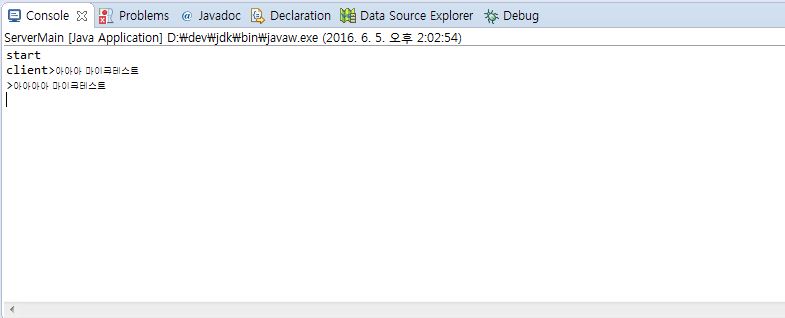


3) Server

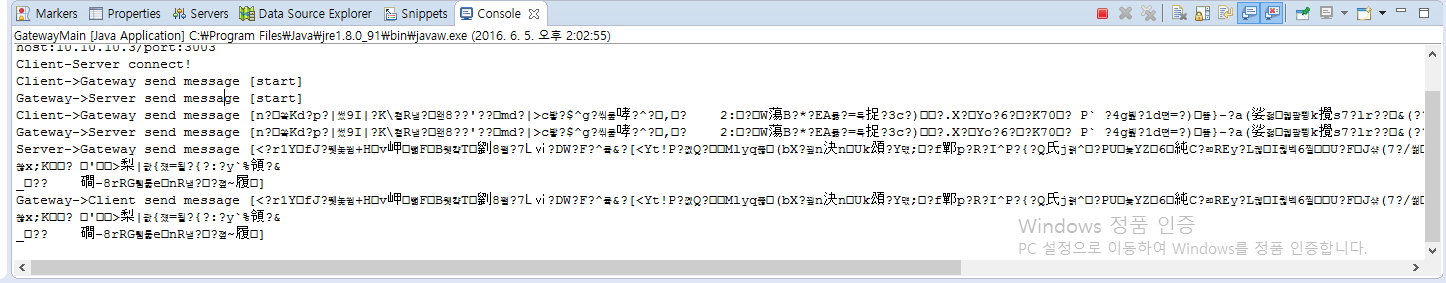


* **데이터전송2 Server → Client**

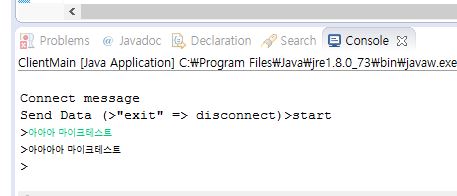
1) Server



2) Gateway

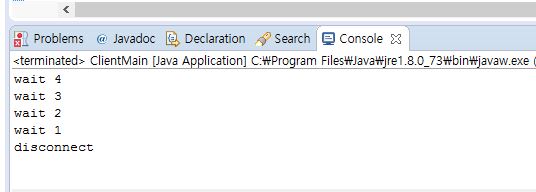


3) Client



* **연결해제** (client가 “exit”을 입력한 경우)

1. Client exit 입력



1. Sever  
   