네트워크 보안 및 실습

Term project

암호화 server - client 통신

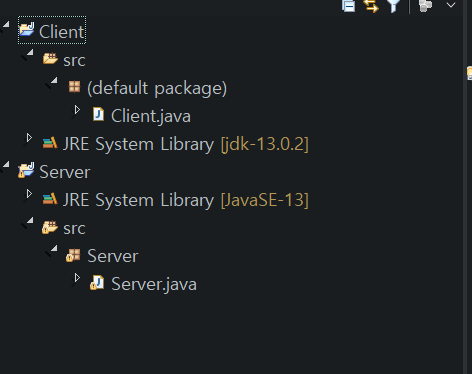
사이버보안학과

201620641

유 상 정

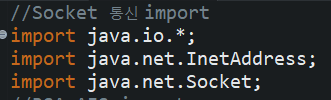
1. 채팅 프로그램 구현

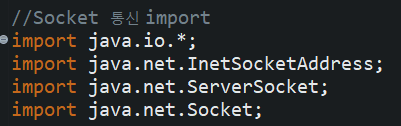
Client-server 구조의 socket 통신을 통해 1:1 채팅 프로그램을 구현하였다. 또한 사용한 언어는 Java Language이고, eclipse를 사용하였다.

* 1. Server, client project 생성

Socket 통신을 이용한 Server – client 구조는 서로 다른 process가 통신하는 것이므로, 별도의 process가 구현되어야 한다. 즉, server, client 각각 project를 생성해야 한다.

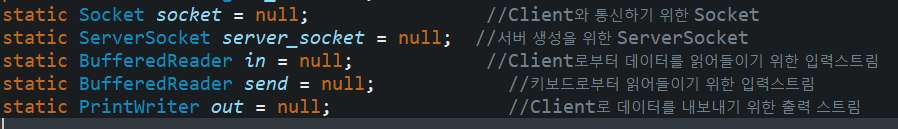
* 1. socket 구현

1. 기본 선언



(server import) (client import)

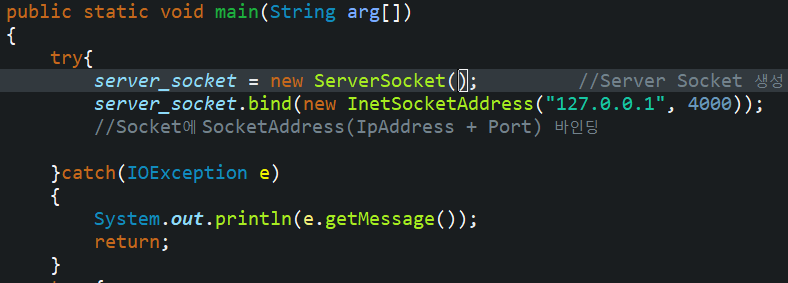
java에서 제공하는 io 및 socket을 import 해준다.



(server의 socket I/O 관련 전역 선언)

Server가 실행되는 동안, 계속 사용될 socket, server\_socket, BufferdReader. PrintWriter를 전역에 선언해준다. Client는 server\_socket을 제외하고 같다.

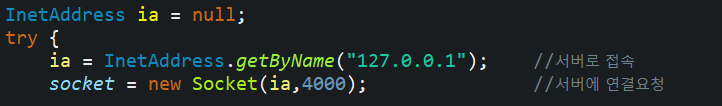
1. Server의 socket 생성 및 bind



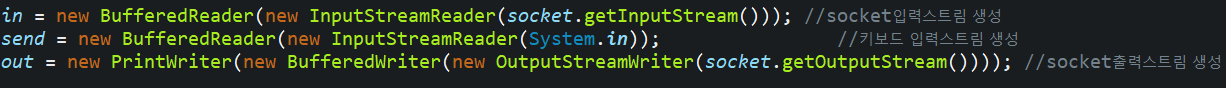
Server의 serverSocket객체를 생성하고, server socket에 ip와 port를 binding 한다.

1. Server의 accept()함수 호출

Server가 accept() 함수를 호출하여, client의 Connect 요청을 기다리고, block 된다.

1. Client의 socket() 함수 호출

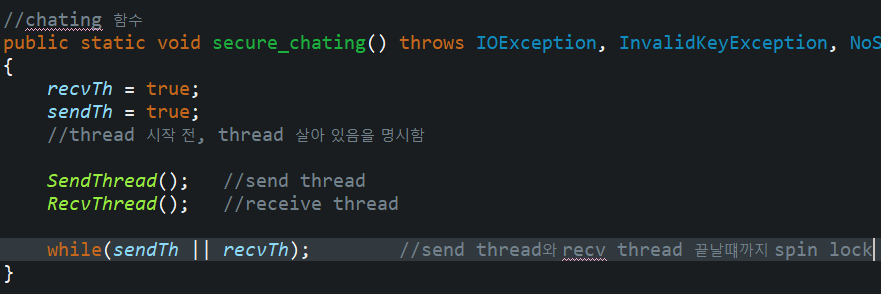
Client가 server의 IP와 port를 이용하여, socket을 생성하고, server에게 connect요청한다. java에서는 connect함수를 사용하지 않아도, socket 객체의 인자로 ip와 port number를 넘겨주면, connect 요청이 자동으로 처리된다.

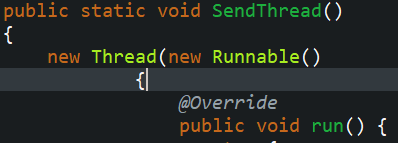
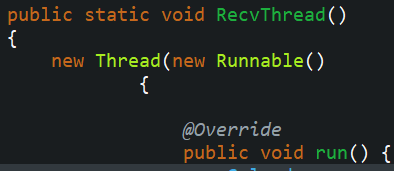
1. Server, client의 입력, 출력 stream 객체 생성

server와 client모두 socket 입/출력 스트림reader, writer와 키보드로부터의 reader 객체를 생성한다.

* 1. Server, Client의 Socket을 통한 채팅

Server와 client 모두 AES 암호화 통신을 사용하는 secure\_chating 함수를 선언한다. 이 루틴 전에, server의 RSA키 쌍 생성, Client의 AES key와 초기벡터(IV) 생성, RSA 암호화와 socket 통한 AES 키와 초기벡터 전달 과정이 이루어진다. 자세한 사항은 암호화 기능 설명부분에서 말한다. chating 함수는 send Thread와 Receive Thread로 구성되어 있다.



위는 secure\_Chating()함수 내용이다. Thread 실행 전, 각 thread가 살아있음을 표시해주고, 실행해준다. 그리고 main함수는 send thread와 receive thread가 끝날 때까지, spin lock 한다. 이 함수는 server와 client가 똑같다.

(send thread) (Receive thread)

1. Send Thread

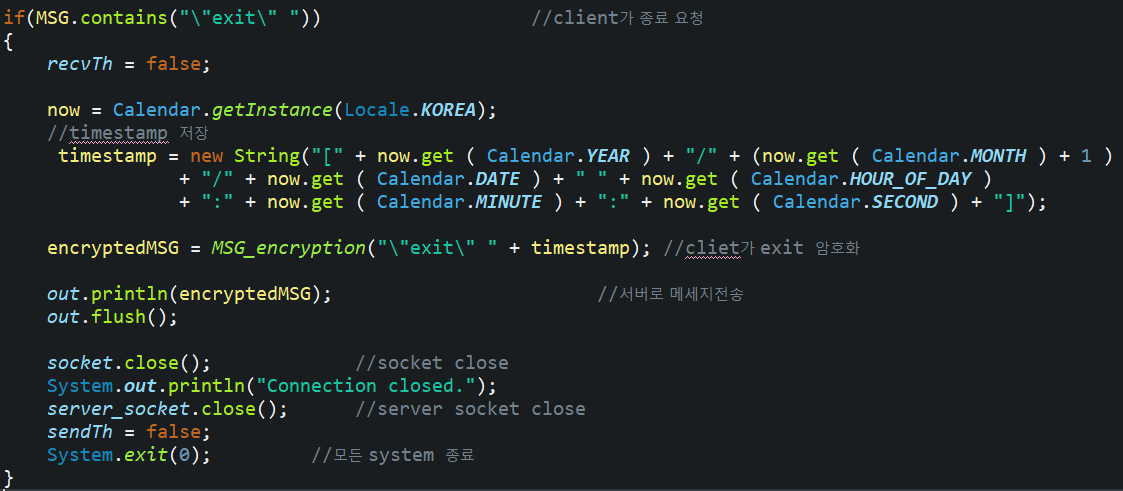


Send thread는 while문으로 구현되어 있으며, rece Thread가 살아있는 동안 계속 된다. 이while문은 사용자로부터 메시지를 입력받고, timestamp와 MSG를 함께 암호화하여, 상대방에게 전달한다. 그리고, 입력한 메시지가 “exit”이면, send Thread가 죽었음을 나타내고 thread를 종료한다. Server와 client 둘다 동일하다.

1. Receive Thread

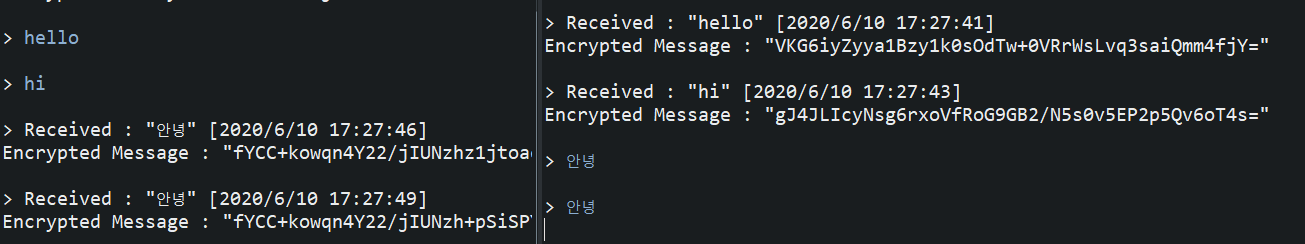


Receive Thread는 while문으로 구현되어 있으며, send Thread가 살아있는 동안 계속된다. 이 while문은 상대방의 메시지를 읽고, MSG를 복호화 하여, 상대방의 MSG와 time stamp를 확인한다.

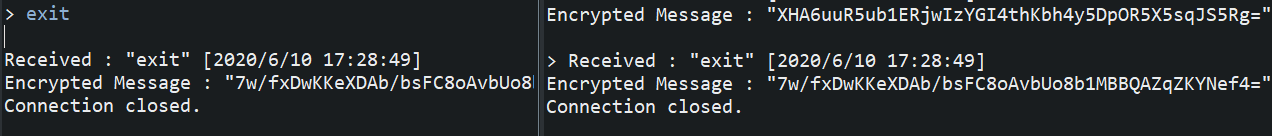


만약 상대방에게 받은 메시지가 “exit”를 포함하면, Receive thread가 죽었음을 표시하고, 상대방에게 timestamp와 “exit”를 암호화하여 함께 전송한다. 그리고. Socket을 닫고, server라면 server\_socket도 닫는다. 그리고, system.exit(0)를 호출하여, 정상적으로 모든 system을 종료한다. Recv Thread도 server와 client가 동일하게 작용한다.

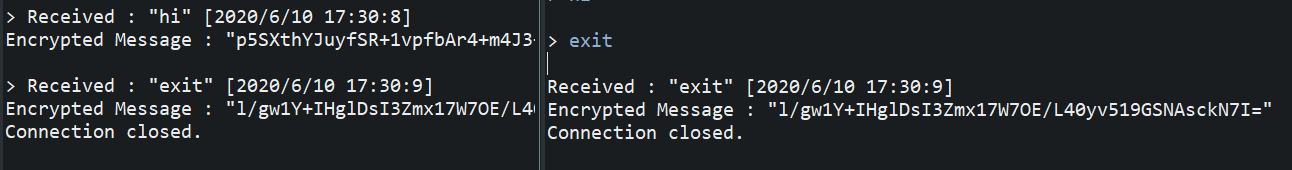
* 1. 실행화면



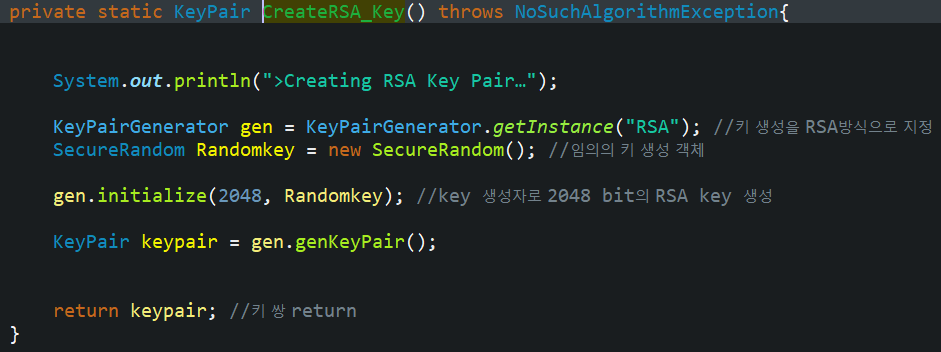
(왼쪽: server, 오른쪽: client)



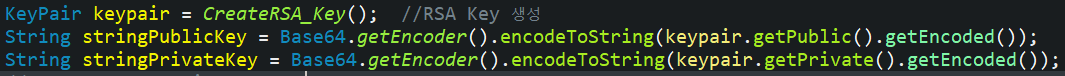
(server가 “exit” 전송)



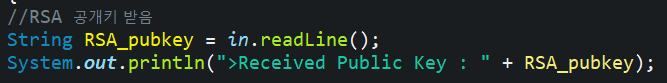
(client가 “exit” 전송)

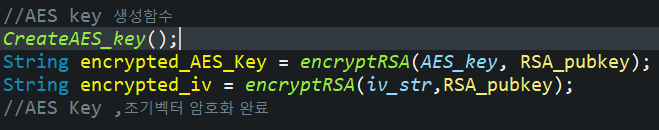
1. RSA를 이용한 키 교환
   1. RSA key생성 및 공개키 전달

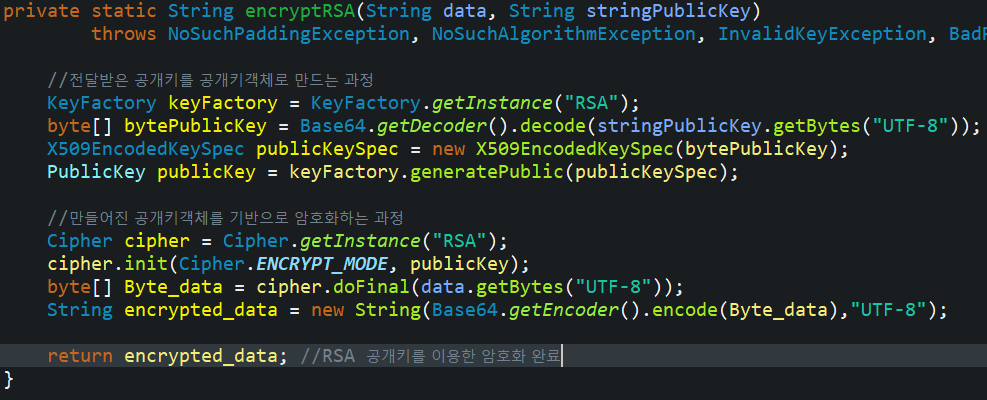
SecureRandom 객체를 이용하여 2048 bit의 random key 쌍을 생성

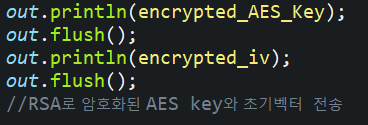
Server에서 RSA의 key 쌍(2048bit)을 생성한다. 이때, KeyPairGenerator와 Secure Random객체를 이용한다.



Server가 key쌍을 string 형으로 전환 후, client에게 공개키를 전달한다. client는 RSA 공개키를 받고, 공개키를 출력한다.

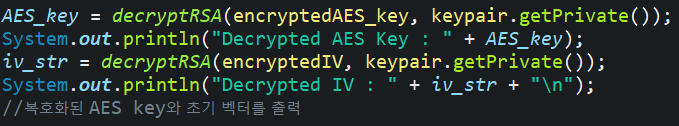
* 1. RSA 공개키를 이용한 암호화

Client가 server로부터 전달받은 RSA 공개키를 이용하여, 생성한 AES key와 초기벡터를 암호화한다. 우측의 함수를 보면, 전달받은 string 공개키를 decoding과정을 거쳐서 PublicKey 객체로 만들고, publicKey를 이용해서 들어온 data를 암호화 후, encoding 한다.

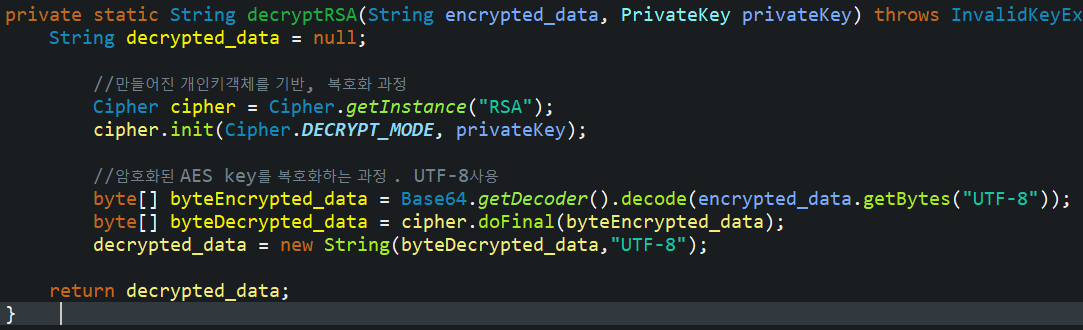
Client는 RSA의 공개키로 암호화한 AES key와 초기벡터를 server에게 전송한다. 만약, 이 내용을 누군가 탈취하여도, 암호화되었기 때문에 실제 key와 초기벡터를 알 수 없다.

* 1. RSA 개인키를 이용한 복호화



Server는 client로부터 암호화된 AES key와 IV를 받는다.

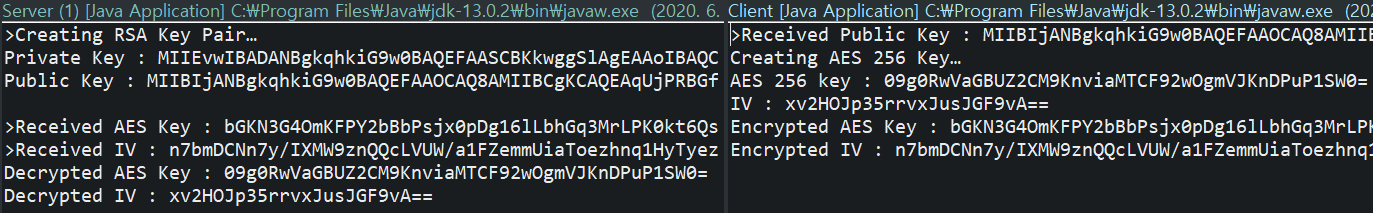
전달받은 AES key와 IV를 RSA 개인키로 복호화 한다.



위는 RSA 복호화 함수이다. 이 함수는 암호화된 data를 decoding 하고, RSA 개인키로 복호화 한다. 그리고 string으로 변환해준다.

2.1, 2.2, 2.3의 과정을 통해서 server와 client가 RSA를 이용한 AES 키 교환에 성공했다.

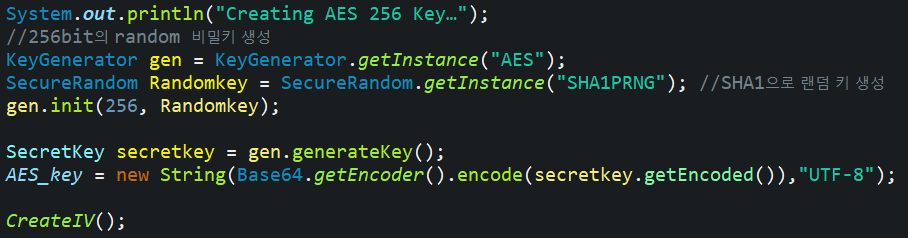
2.4. 실행화면



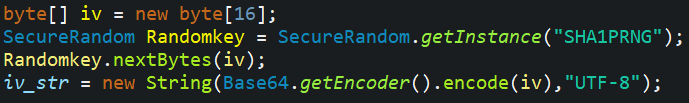
왼쪽은 server이고, 오른쪽은 client이다. 위의 사진을 보면 server가 생성한 RSA 공개키를 client가 받았고, client는 RSA공개키를 이용해 AES 256키와 초기벡터를 server에게 암호화하여 전송한다. 그리고 server는 RSA개인키를 이용하여 받은 키와 초기벡터를 복호화 한다. 위를 보면 Client가 생성한 AES 256 key와 IV가 server에서 Decrypted된 AES key, IV와 같음을 알 수 있다. 즉, RSA를 이용한 키교환이 성공적으로 이루어졌다.

* 참고
* RSA는 비대칭 키로서 비밀키와 공개키가 존재한다. 또한 공개키는 누구나 알 수 있지만, 비밀키는 생성자만 알 수 있기 때문에, 키 교환에 적절한 암호시스템이다.

1. AES 암호화 통신
   1. Client의 AES key와 IV 생성

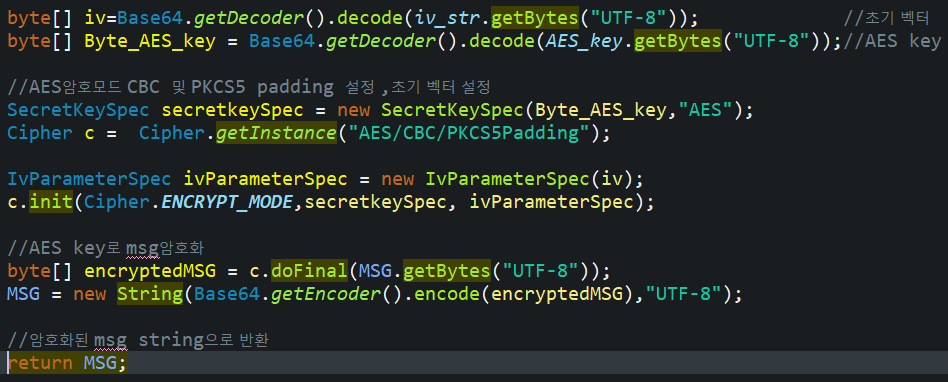


위의 함수는 AES key를 생성한다. RSA 키 생성처럼 KeyGenerator와 SecureRandom을 사용하며, 256bit의 AES key를 생성한다.



위의 함수는 초기 벡터를 생성하는 함수로서, SecureRandom을 사용하여 128bit의 random한 초기 벡터를 생성한다.

* 1. AES key를 이용한 메시지 암/복호화



위의 함수는 AES key를 이용한 메시지 암호화 함수이다. IV와 AES key를 설정하고, 암호모드를 AES/CBC/PKCS5Padding으로 설정해준다. 그리고 메시지를 암호화한다.



위의 함수는 AES key를 이용한 메시지 복호화 함수이다. IV와 AES key를 설정하고, 암호모드를 AES/CBC/PKCS5Padding으로 설정해준다. 그리고 메시지를 복호화 한다.

* 1. 실행화면

위의 사진을 보면 client가 보낸 “안녕” 이 AES key로 암호화되어 server에게 전달되었고, server는 AES key를 통한 복호화를 통해 “안녕”을 성공적으로 얻어냈다.

* 참고
* AES는 대칭 키암호로서, 메시지 암/복호화를 위해서는 키 교환이 이루어 져야한다. 그리고 키 교환에 성공하면, 동일한 키를 이용해서 메시지 암/복호화를 할 수 있다.

1. 결론

위에서 설명한 일련의 과정을 통해서 Server-Client구조의 1:1 암호화 채팅프로그램을 구현하였다. 이론적으로만 알고 있던 RSA와 AES를 구현한 결과를 확인하니 실제로, 원래의 data를 알아볼 수 없는 수준으로 암호화됐다. 이는 중간에 누군가가 메시지를 탈취하여도 그 내용을 알 수 없으므로, 암호화 통신의 중요성을 다시 깨닫게 되었다. 또한 비대칭키 암호와 대칭 키 암호화를 모두 사용함으로써 효율적으로 프로그램을 구성할 수 있었다. 왜냐하면, 이론적으로 대칭 키 암호는 속도가 빠르지만, 키 교환의 문제가 존재하고, 비대칭 키 암호는 속도는 느리지만, 키 교환의 문제가 없기 때문이다.

이 프로젝트를 행하면서 암/복호화에서의 인코딩, 디코딩의 중요성을 깨달았다. 왜냐하면, 암/복호화에 사용되는 것은 byte단위이지만, 실제로 우리가 이해할 수 있는 것은 string 단위이다. 즉, 암/복호화에서 적절한 인코딩, 디코딩은 필수적이라고 할 수 있다. (실제로, project를 진행하면서 UTF-8 방식을 빠트리고 진행하여, 한글 깨짐 현상이 발생하였다.) 이러한 기능을 java에서 Base64로 제공하여 설계자가 손쉽게 할 수 있다. 그래서 이번 프로젝트에서 Base64의 인코딩, 디코딩을 UTF-8 사용하여, 까다로운 한글도 문제없이, 암호화 통신에 성공하였다.

프로젝트를 하면서 다뤄보지 않은 암/복호화, 인코딩/디코딩을 진행하면서 보안 프로그래밍을 수행하면서 실습에서 진행했던 과제와는 다르게, 완성형 프로그램을 제작함으로써, 많은 고민을 해야 했다. 하지만, Java에서 제공하는 편리한 기능 덕에, 프로그램을 만드는 시간은 그리 오래 걸리지 않았다. 왜냐하면 java에서는 암호화와 관련된 거의 모든 기능을 import를 통해서 얻어올 수 있었다. 이 project를 수행하면서 java는 암/복호화에 최적화된 Language가 아닐까 라는 생각을 해보았다. 본인은 이번 project를 통해 익힌 RSA와 AES의 암/복호화 기능을 다른 프로그램에도 적용하여, 보안 프로그래밍 실력을 좀 더 향상시킬 예정이다. 개인적으로 부족한점은 java에 대한 이해도 부족이 가장 컸던 것 같다. 왜냐하면 송신 thread와 수신 thread 및 socket 통신 같은 경우, C에서 해보았기에, 쉽게 생각하였다. 그러나 개념은 이해했으나, java의 구현법을 잘 몰라서 어려움을 많이 겪었다. 특히, thread구현에서 가장 큰 어려움을 겪은 것 같다. Java에 대해서 깊게 공부를 할 예정이다.

1. 참고 자료

* <https://lktprogrammer.tistory.com/62>

Java에서의 server, client 통신 구현

* <https://offbyone.tistory.com/346>

RSA 암/복호화 참조

* <https://this-programmer.com/entry/Java%EC%9E%90%EB%B0%94Spring%EC%8A%A4%ED%94%84%EB%A7%81-RSA-%EA%B3%B5%EA%B0%9C%ED%82%A4-%EA%B0%9C%EC%9D%B8%ED%82%A4-%EC%95%94%ED%98%B8%ED%99%94-%EB%B0%A9%EC%8B%9D%EC%9D%84-String%ED%98%95%ED%83%9C%EB%A1%9C-%EB%8B%A4%EB%A4%84%EB%B3%B4%EC%9E%90>

PublicKey, PrivateKey 객체를 string형으로 바꾸는 방법 참조

* <http://blog.naver.com/PostView.nhn?blogId=wpdus2694&logNo=220898932452&redirect=Dlog&widgetTypeCall=true>

java의 AES 암/복호화 참조

* <https://dev-syhy.tistory.com/15>

base64 인코딩, 디코딩 참조

* <https://jang8584.tistory.com/232>

java 날짜 반환 참조

<https://selfdevelope.tistory.com/173?category=714556>

java.security.InvalidKeyException: Parameters missing error 해결 참조

* <https://blog.naver.com/PostView.nhn?blogId=ilikebigmac&logNo=221587381568>

java에서 thread 사용한 채팅 예제 참조

* <https://coding-factory.tistory.com/526>

java 모든 system 종료 참조

* <https://fruitdev.tistory.com/72>

java 문자열 포함 여부 확인 참조