# CS 6238 – Project 2Report

**Done by: Sunny Neo and Yi Ding**

Contents

[CS 6238 – Project 2Report 1](#_Toc405243077)

[1. Design 3](#_Toc405243078)

[1.1. Class Documentation 3](#_Toc405243079)

[1.1.1. Server.java 3](#_Toc405243080)

[1.1.2. Client.java 3](#_Toc405243081)

[1.1.3. Document.java 3](#_Toc405243082)

[1.1.4. DelegationToken.java 3](#_Toc405243083)

[1.1.5. Commands.java 3](#_Toc405243084)

[1.1.6. Util.java 3](#_Toc405243085)

[2. Implementation 4](#_Toc405243086)

[2.1. HandShake 4](#_Toc405243087)

[2.2. Sending of Delegation Tokens 5](#_Toc405243088)

[2.3. GET FUNCTION 6](#_Toc405243089)

[2.4. PUT FUNCTION 7](#_Toc405243090)

[3. Test Case 8](#_Toc405243091)

[4. Tasks Completed 16](#_Toc405243092)

[5. Contribution 16](#_Toc405243093)

# Design

The design of this project is started with the bottom up approach whereby we identified all the necessary building blocks of the system such as the socket programming, cryptography modules and utility functions that are required and aid us with the development of different components of the system. From there, we started looking at the classes that will hold our data namely history file and instruction tables.

Once we are done with all the primitive functions and classes, we split the project into three major processes such as the initialization phase (handshake) to get shared secret key, put & get functions and delegation functions and started thinking how we could put all the previous blocks together in these processes.

# Class Documentation

## Server.java

Server.java is the class for all operation pertaining to Server’s functionality. It is a multiple threaded socket that can handle multiple clients at one time.

## Client.java

Client.java is the class for all operation pertaining to Client’s functionality. It connects to the Server and run a socket for receiving delegation tokens from other clients.

## Document.java

Document is the class to use to wrap Meta data information about the file uploaded by clients. The security property of data is based on the client’s configuration upon uploading it to the server.

## DelegationToken.java

DelegationToken is the class used to hold Delegation Token information. The client who provides token will be granted permission for respective files.

## Commands.java

Command.java is the class used by the Client & Server for communicating functions to be executed.

## Util.java

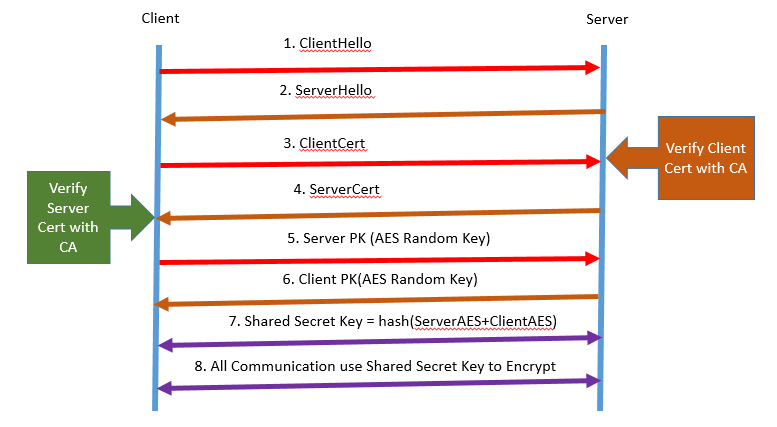
Util.java is a utility class that provides cryptography operations such as symmetric and asymmetric encryption, apart from that, it provides file IO operations and serialization.

# Implementation

The protocol used for network is TCP sockets. The Symmetric encryption used is AES/CBC/PKCS5PADDING. The Public-Private Key are generated based on RSA algorithm using Keytool from Java. The certificates of server and clients are signed by self-signed CA created using Openssl.

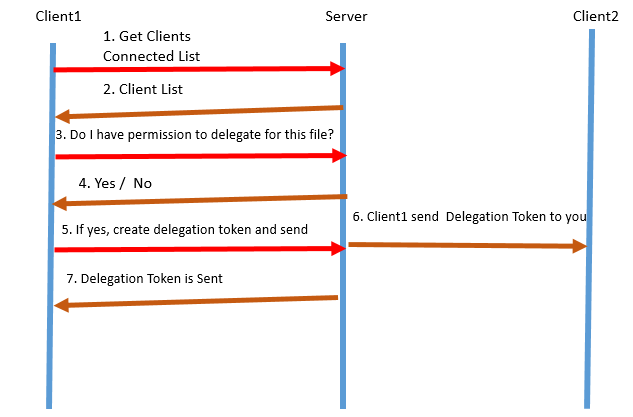
# HandShake

Protocol for HandShake between Server and Client to establish Secure Channel



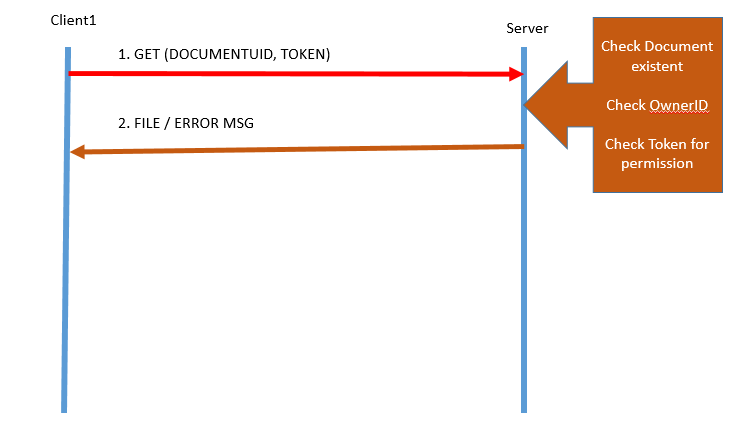
Upon receiving the certificates, the client and server will verify the public key of each other against the trusted CA stored in both of them. If the verification fails, the handshake will be terminated. Once the handshake is completed, all communications are encrypted using the shared secret key.

# Sending of Delegation Tokens



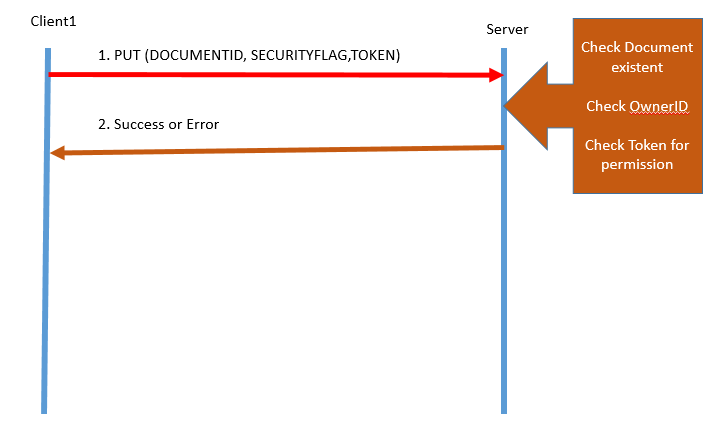
The protocol for sending delegation token is illustrated above. The clients do not have directly communication among themselves, delegation token is created by client and send to the server, which in turn will be forwarded to the respective recipient. Therefore in our design, the designated user must be connected to the server. All communications between clients and server are secured using the respective session keys created during handshake for each client-server pair.

# GET FUNCTION



When the Server receives a GET command from Client, he will check whether the client is the owner of the file, if it is not, it will continue to check for valid delegation token with the valid rights, then it would check the meta data of the document to verify the document based on the security flag that it has been created(confidentially, integrity, none) before allowing/denying the client from getting the file.

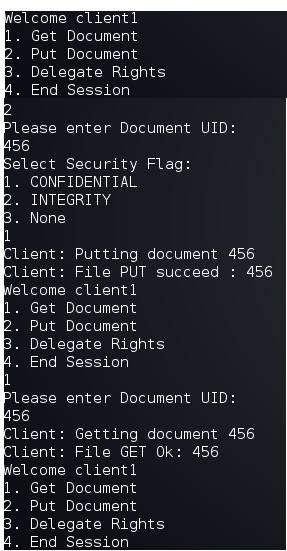
# PUT FUNCTION



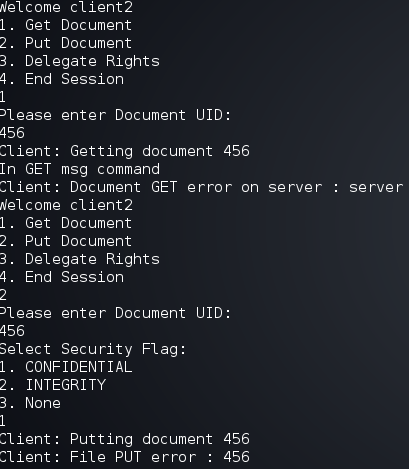
When the Server receives a PUT command from Client, he will check whether the client is the owner of the file, if it is not, it will continue to check for valid delegation token with the valid rights before allowing/denying the client from putting the file. If the file exists, after uploading a new file by other clients, the owner of the file remains

# Test Case

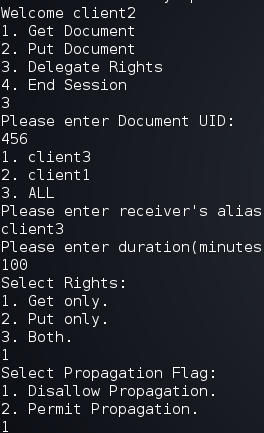
* 1. **Positive: client1 put a new file on the server and get file on the server**



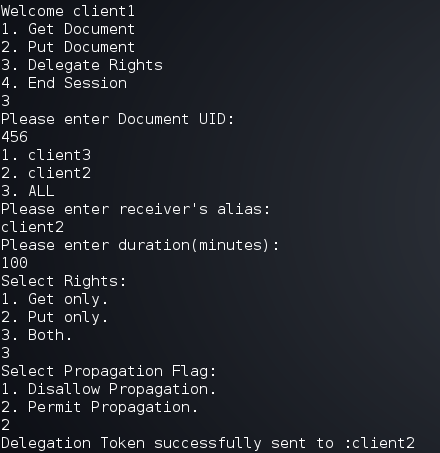
* 1. **Negative: client2 tries to get a file created by client1**



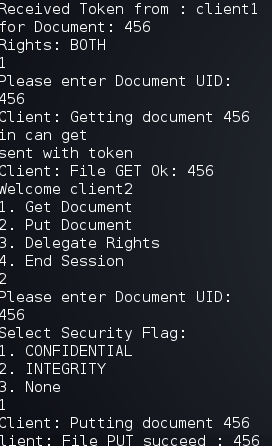
* 1. **Negative: Client2 tries to create a delegate token which he has no rights to**



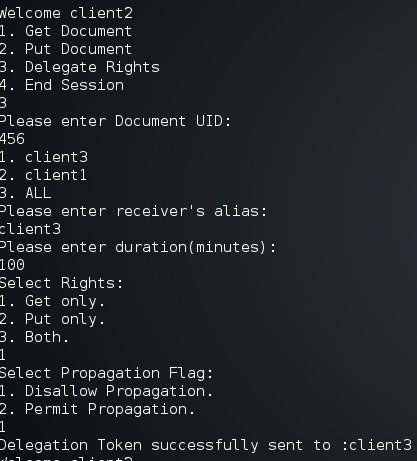
* 1. **Positive: client1 creates delegation token for client2 and send it to him via the Server**



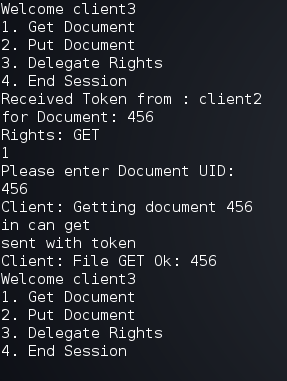
* 1. **Positive: After client2 received the token, he has rights to get and put 456**

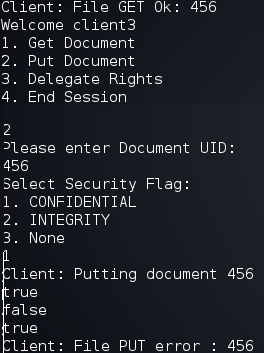


* 1. **Positive: client2 creates delegation token for 456**

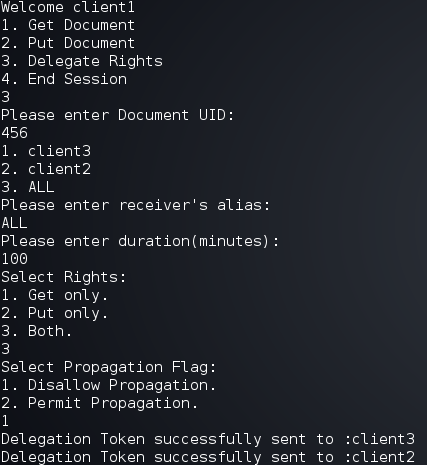


* 1. **Positive/Negative: after client3 received the delegation token from client2, he is able to get document 456 but he’s not able to put 456**



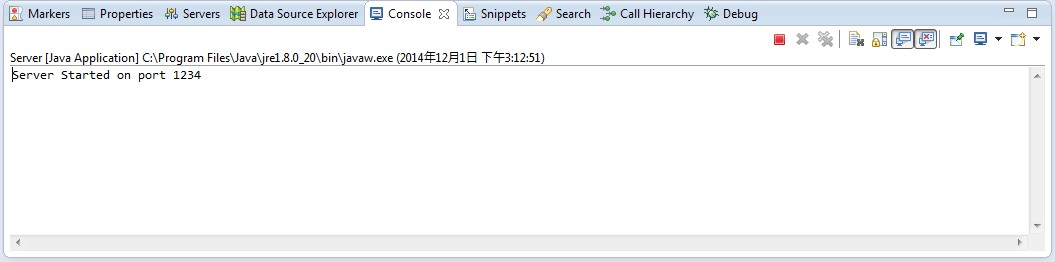


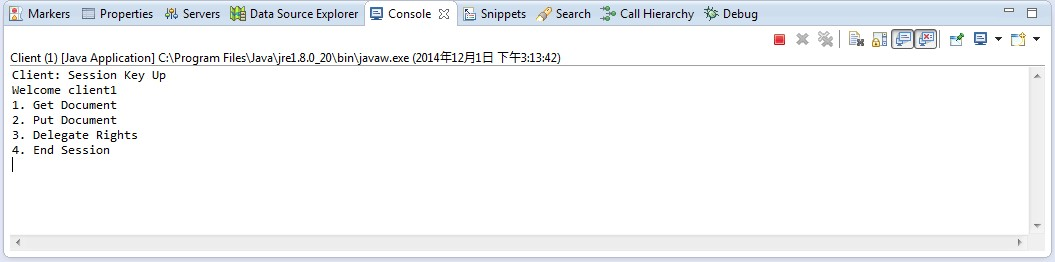
* 1. **Positive: Creating token for ALL users connected – restart program**

****

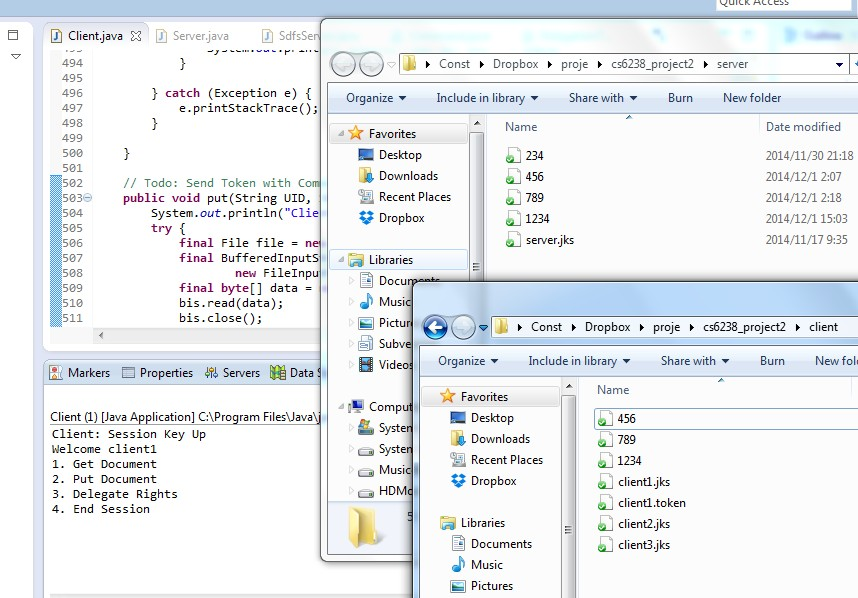
## Other Screenshots of Working Application

**Server started**

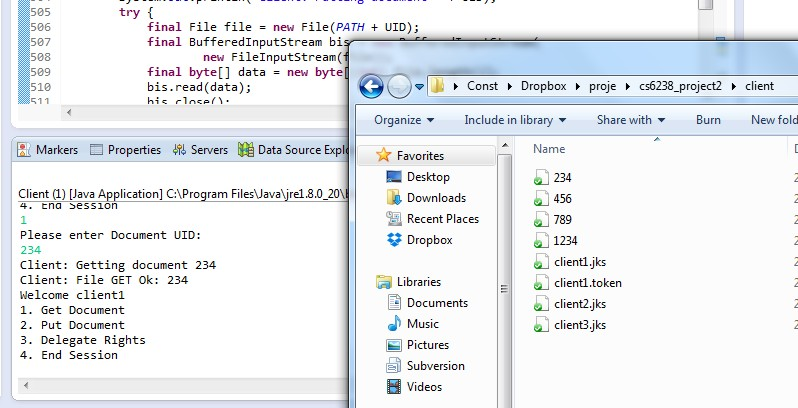




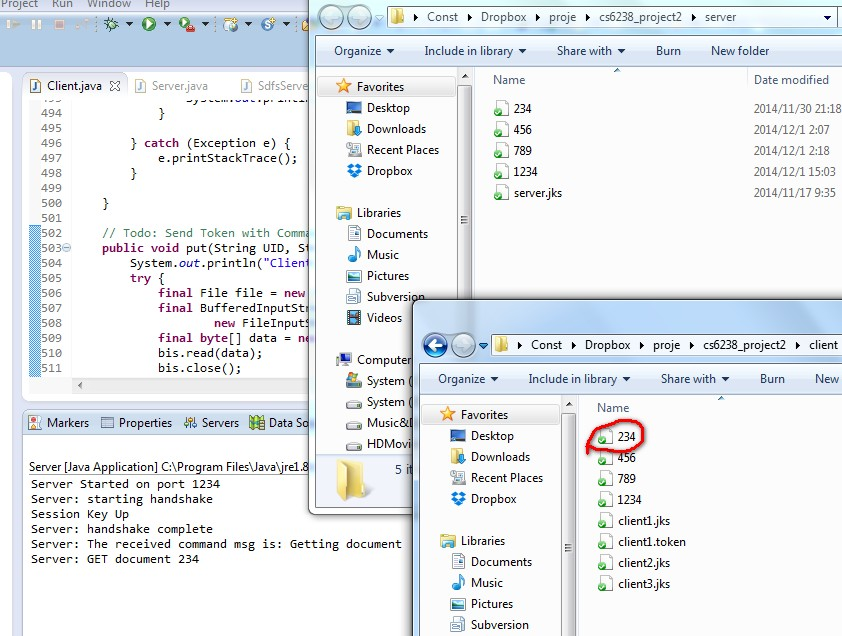
**Client1 is up**



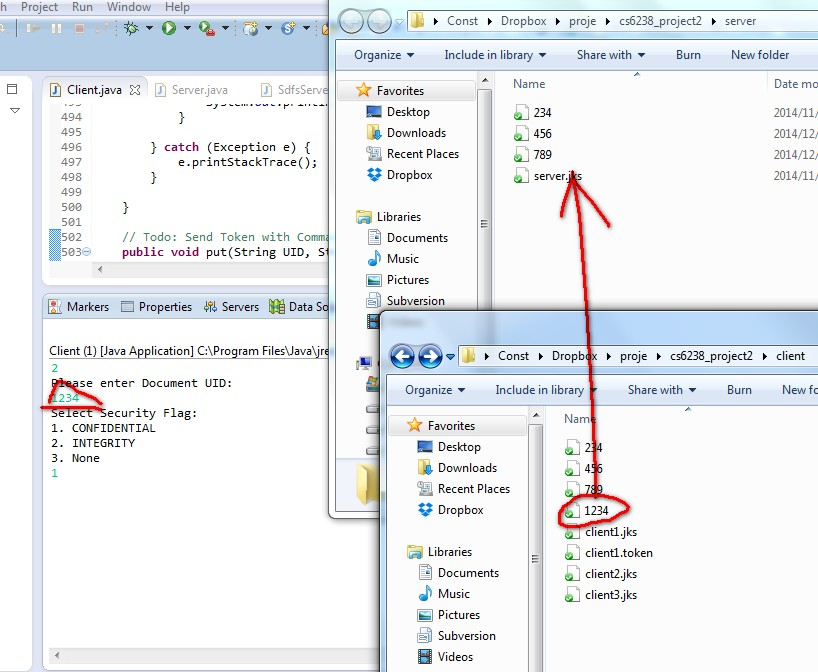
**Client before GET FILE 234**



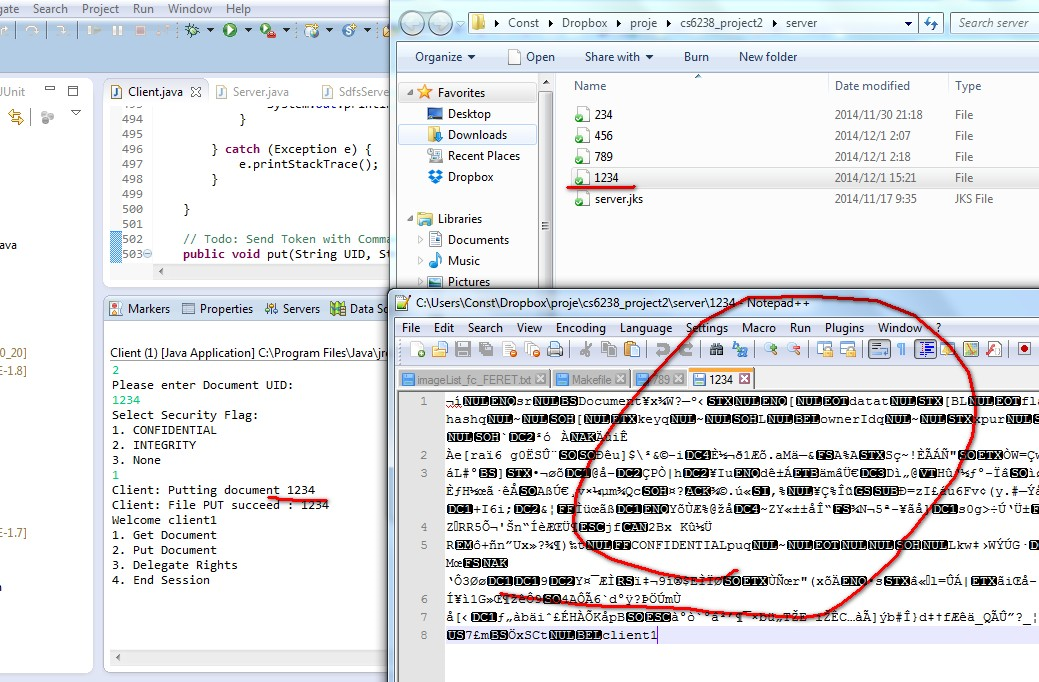
**Client After GET FILE 234**



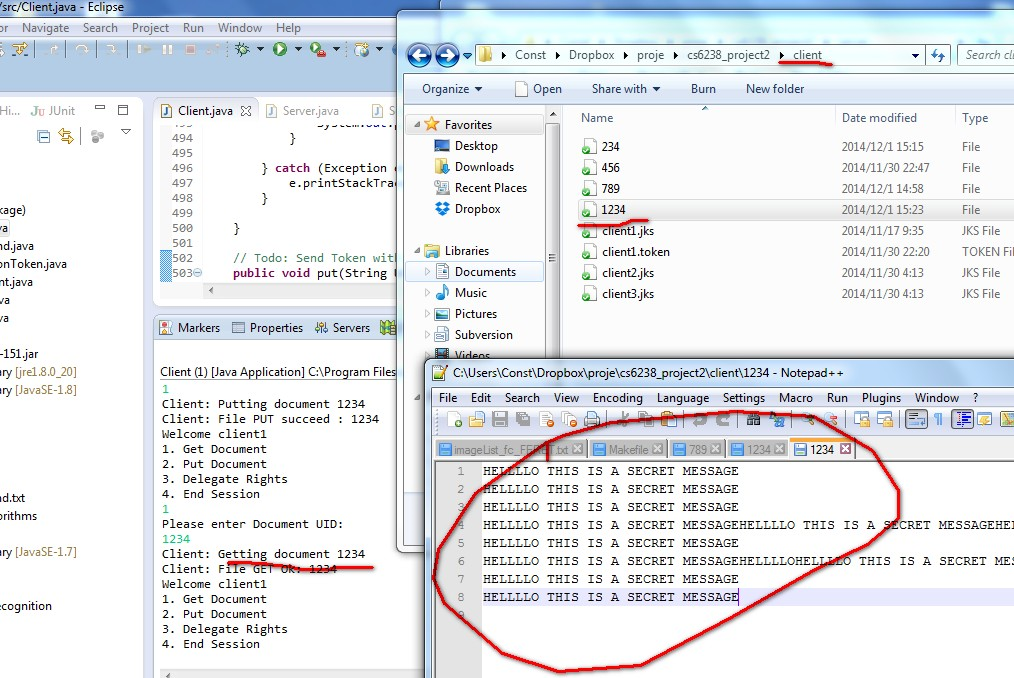
**Before Client put a new file into Server 1234**



**After Client put 1234 file into Server**



**Get file 1234 again, decrypted**



# Tasks Completed

* 1. Start Session – Hand Shake are written from scratch and a session key is created for every server-client pair communication.
  2. Get Function – Checking of owner and delegate token and verifying the security of the document based on the meta data security flag that was defined before granting permission.
  3. Put Function – Checking of owner and delegate token before granting permission. In addition to store the file based on the security flag, confidentiality for encrypting the data with a random key encrypted by the server’s public key, integrity signed with document key and verify the signature before sending it, and no encryption for None.
  4. Delegate – Checking of owner and delegate token before granting permission, before creating new delegation token. Creating delegation token for “ALL” connected clients or per client, once created, they will be sent to the Server and the designated clients will receive it via the server via secure channel (server-client shared secret key pair).
  5. End Session – End the session of the client

# Contribution

Sunny – Handshake, Util Class, 50% of delegation

Yi Ding – Put & Get Function, commands and 50 % of delegation.

All in all, we split the work equally.