**FILE TRANSFER PROJECT DOCUMENTATION**

By

BAO NGUYEN – BCN140030

MOFAN LI – MXL162930

CS 6349.001

**I. Project Description and Features**

*1. Description*

The problem is to create a file transfer system that utilizes Socket Programming, Connection Management, Reliable Communication, and security protocol that utilizes SHA-1. It allows 2 individuals (1 server and 1 client) on different computers to communicate via network. All messages transported over the network shall be secured with respect to Authentication, Confidentiality, and Integrity.

*2. Features*

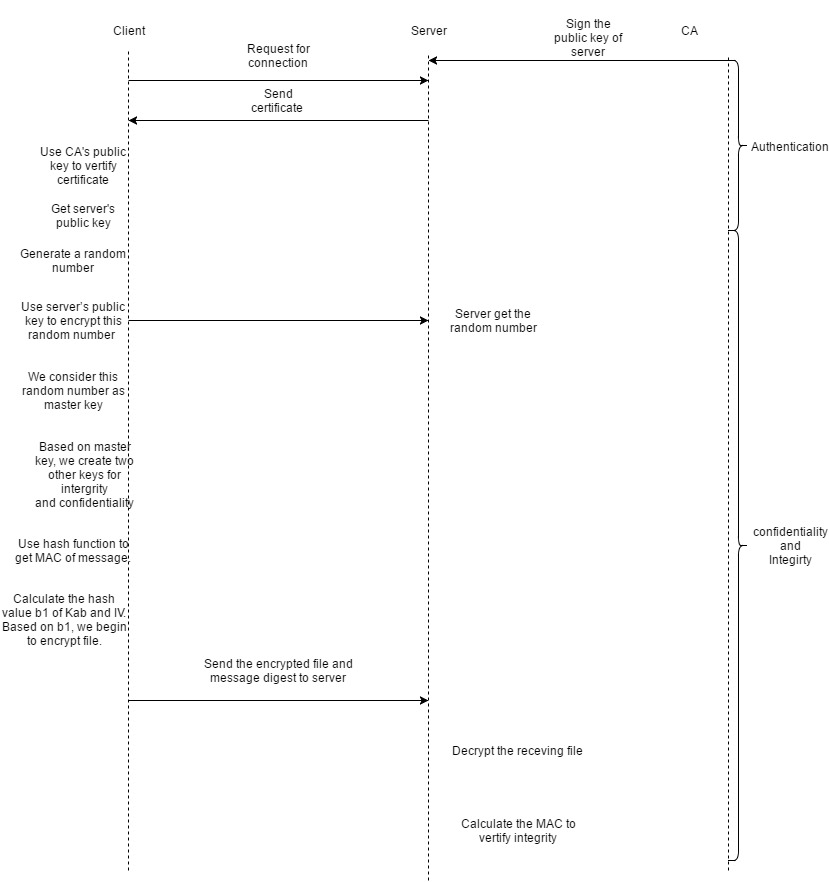
We use Java to build this program. We create 2 different projects/modules, a *Client.java* and a *Server.java*, both lie within the *FileTransferApplication* project. The server sets up the sockets and wait for connection. The client sets up its socket and connect to the server. This program also records the start and end time of the session. This project supports Windows, Mac, and Linux.

The server does not initiate any message. It waits for the commands from the client and sends responses. The client is able to view the files it contains, to request server to view all the files the server contains, to download files from the server, or to upload files from local host to store in the server.

Please read the included READ\_ME.txt file for more information about executing the program.

**II. Implementation**

*2. Security Protocol*

**

**IV. Detail about Security Protocol**

*1. Authentication*

Server has a certificate which is its public key signed by CA’s private key. The client has CA’s public key. When server receiving the connection request from client, server will send the certificate to client. Client uses CA’s public key to verify the certificate and get server’s public key. After authentication, client generates a random number. We consider this random number as a master key.

Encrypt master key with server’s public key and send it to server.

*2. Confidentiality*

In this project, we use Sha1 to encrypt our files which is generating and using one-time pass to XOR with data.

b1 =Sha1(Kab|IV) c1=p1⊕b1

b2 =Sha1(Kab|c1) c2=p2⊕b2

… …

bi =Sha1(Kab|ci-1) ci=pi⊕bi

*3. Integrity*

We also use Sha1 to calculate the MAC(Message authentication codes).

MAC =SHA1(Kab|m)

**V. Threat Models**

*1. Eavesdrop*

Even if attacker can watch the message between client and server pass over the network, the attacker still can’t learn the contents of message between client and server because they are encrypted with 1-time pad keys.

*2. Initiate a Conversation Pretending to be Client*

The project has one-way authentication, so the server cannot verify the client. It assumes the first client who successfully connects is the legitimate client.

*3. Read Server’s Database*

The attacker can break into server’s machine and get its private key to decrypt the master key from client. But since each session has a new key, and the master key is not used for encryption/decryption, stealing server’s private key does not help attacker to decrypt the captured messages.

*4. Man-in-the-Middle Attack*

The attacker cannot get useful information if he doesn’t know all the keys (server’s private key and the unique sessions’ keys).

The attacker can steal a message and does not forward to a party, in which case after sometimes both parties will end the connection due to timeout. Even though each party cannot do anything to prevent such an event, they still can detect the message has been stolen.

The attacker cannot successfully execute replay attack because each message is marked with a sequence number. If the receiving message does not have the expected sequence number, it will be dropped by the recipient. Intrusion detection will notifies the client/server after receiving more than 5 out-of-sync messages, which causes them to terminate the connection.

The attacker cannot tamper with the message without being detected because each message is appended with a checksum and is signed by the sender.