SENG360 A3 Technical Manual

2017-11-09

Jory Anderson	V00843894
Purvika Dutt	V00849852
Tim Salomonsson	V00807959

Compilation

(Use with Java 8 or lower)

The submitted zip file contains :

1) A directory of all source files :
    - Server.java
    - ServerMsgGUI.java
    - Client.java
    - ClientMsgGUI.java
    - EncryptedMessage.java
    - KeyPairGen.java
    - Message.java
    - SymKeyGen.java
    - UserDB.java
    - Keystores (a directory containing key pairs generated during compilation)

2) Technical Manual (this file)

The first step to be accomplished to run these files is by compiling all the java files in the zip. This can be done using the following steps:

1) Extract the zip file and open a console inside the included folder.
2) You can compile all the java files in the directory using the command “javac *.java”.

Credentials

Default Usernames and Passwords:

1. Server: admin, adminpassword
2. Client: user, userpassword

Program Usage

(See **Compilation** before proceeding)

1. Using Terminal, navigate to the folder containing the .class files and type 'java KeyPairGen' into the command-line. This will generate a folder containing keyStores for Client and Server. The folder is located in the directory you ran KeyPairGen.

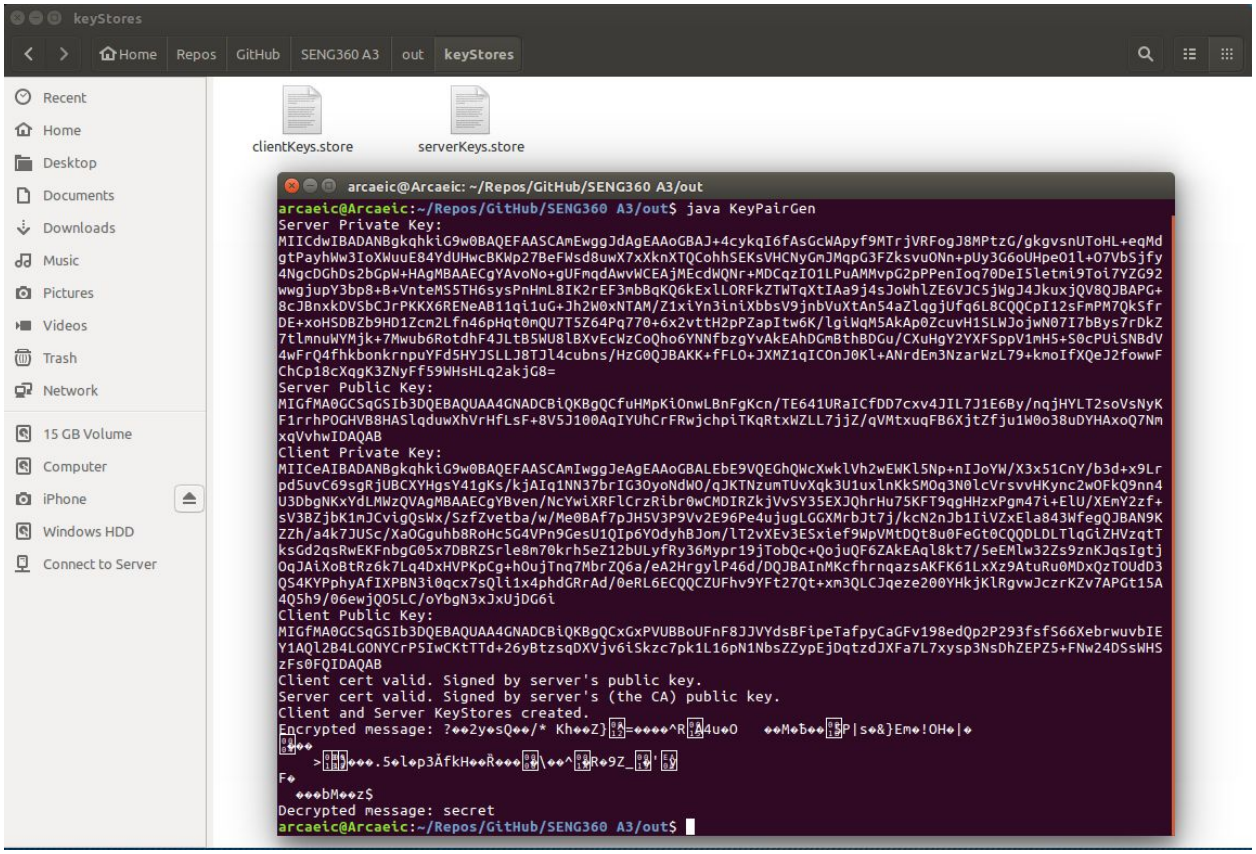

Photo: Generating KeyPairs

2. Next, type 'java ServerMsgGUI' into the command-line. At the bottom, check which parameters you want enabled, enter a username and password if required, and press confirm.
    a. Confidentiality:
        i. Encrypts the message at each end-point.
    b. Integrity:
        i. Ensures the message has not been tampered with by validating the associated Message Authentication Code.
    c. Authentication:
        i. If Authentication is enabled, the "Username" and "Password" fields will become fill-able. Entering a login will generate a hash of the password using the stored salt, and attempt to match the hash+salt with a user password in the UserDB.
        ii. Also, mutual authentication through signed certificates. Server acts as CA and signs all client certificates. Automatically configured through execution of KeyPairGen.main().

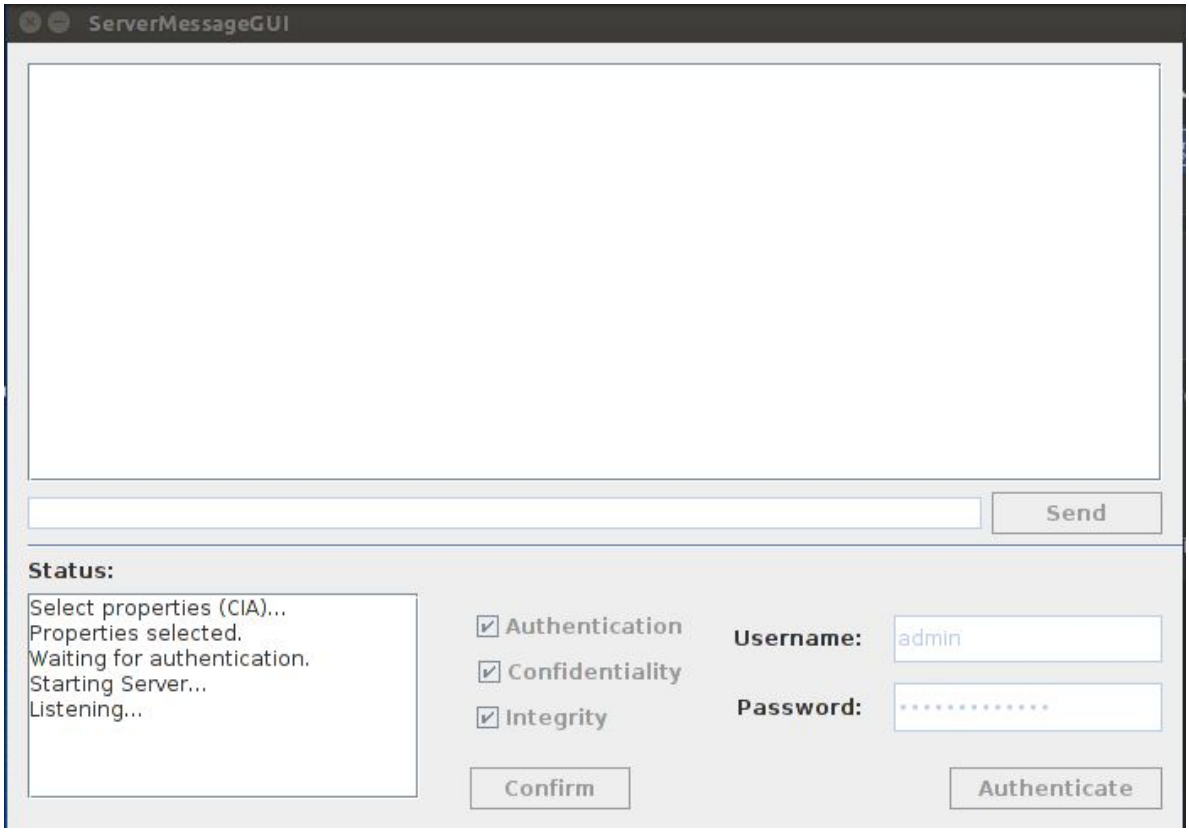

Photo: Inserting Server Parameters

3. Next, type 'java ClientMsgGUI' into the command-line. A UI window identical to Step 2 will open. Do the same as you did for the Server instance. Your client parameters must match the server parameters (See step four).

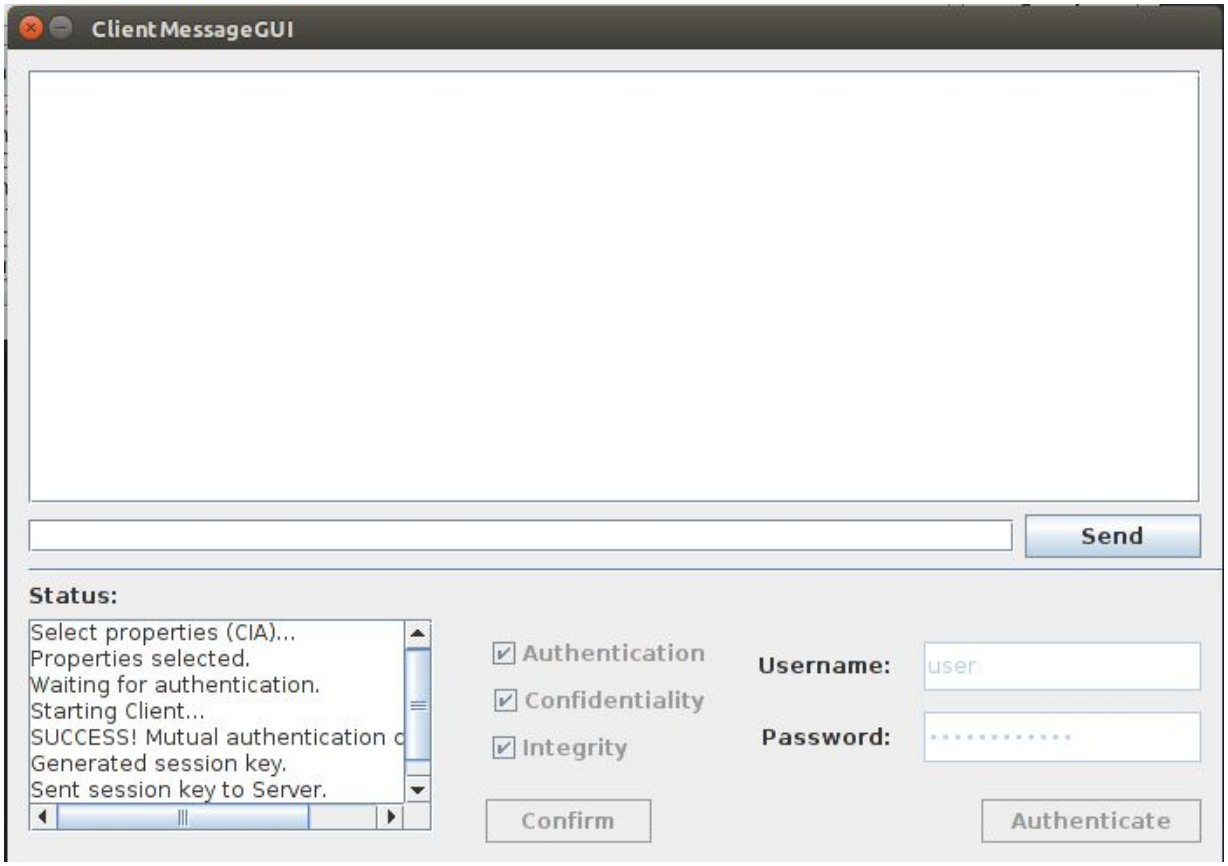

Photo: Inserting Client Parameters

4. If the list of parameters entered by the Server and Client match, a connection will be established and the Server and Client can freely exchange instant messages.
    a. If the Client process is terminated, the Server will begin listening for a new client.
    b. If the parameters do not match, the Client process is closed.

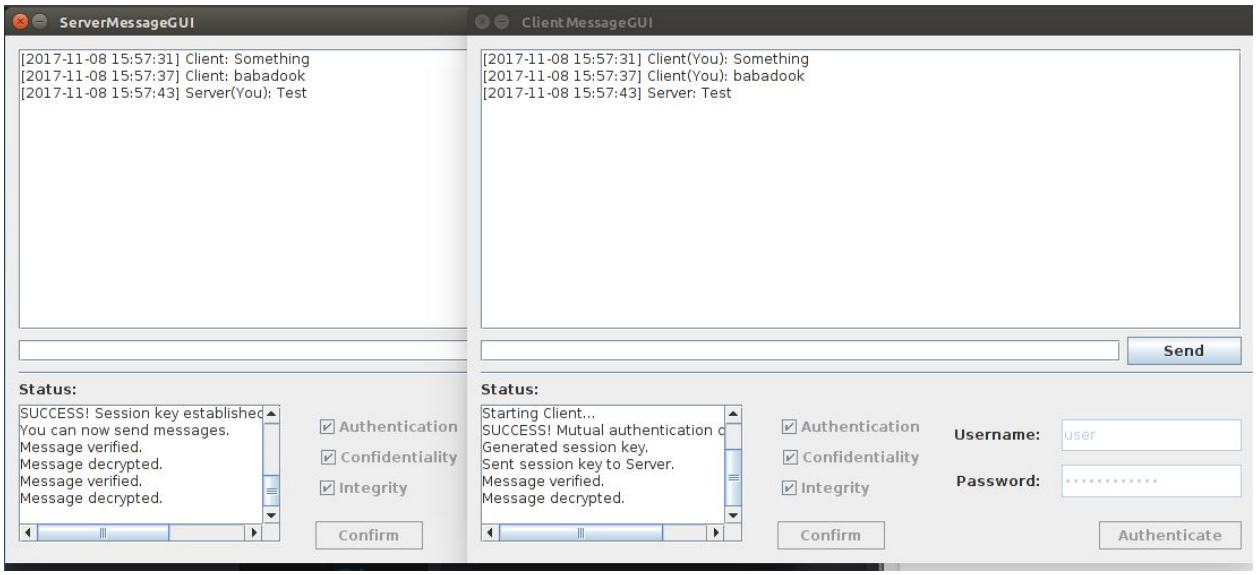

Photo: Server and Client Parameters Match

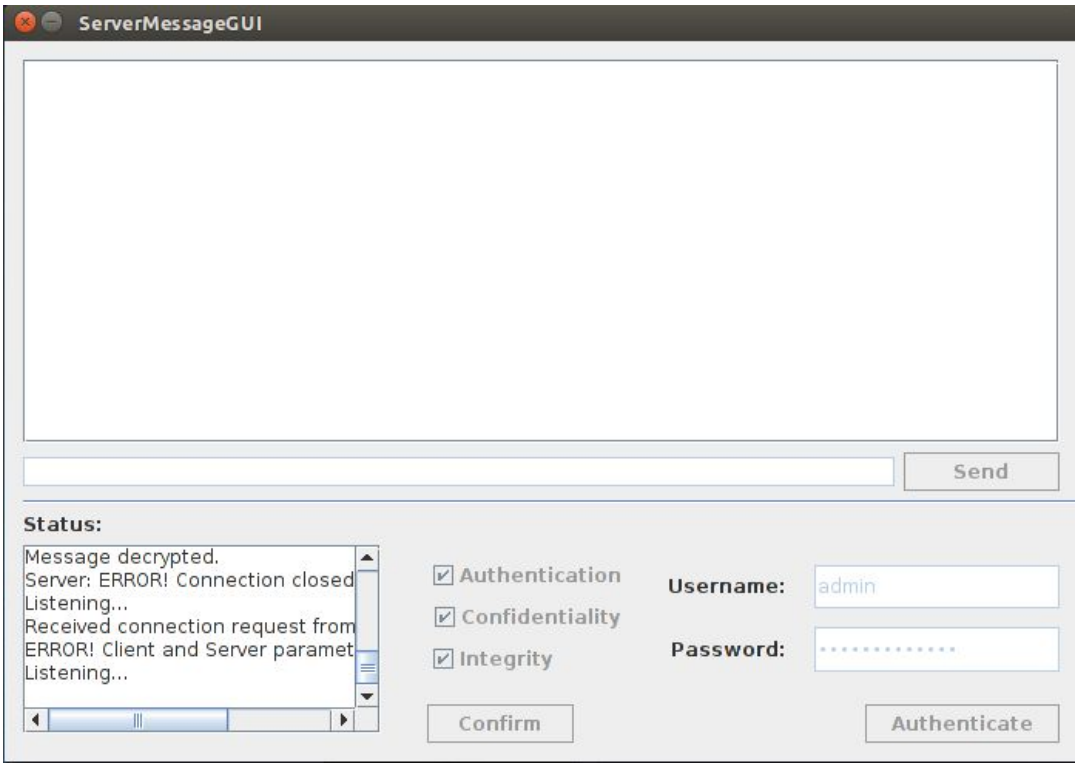

Photo: Mismatch between Client and Server Parameters

Technical Implementation Details

KeyStores/Key Pairs/Signed Certificates

The assignment requires private keys, pre shared public keys and signed certificates. Execution of KeyPairGen.main() will generate and populate two KeyStores in the /keyStores folder, one for the server and one for the client. The KeyStores will contain:

A. Server: "/keyStores/serverKeys.store"
    a. Server's private key
    b. Server's signed certificate
    c. Client's signed certificate

B. Client: "/keyStores/clientKeys.store"
    a. Client's private key
    b. Client's signed certificate
    c. Server's signed certificate

For the purposes of this assignment, the server will act as the Certificate Authority (CA). This means the server will sign its own certificate to create a root certificate. The client's

certificate is then signed with the root certificate. The signed certificates are then stored in the KeyStores.

Session Key Establishment

If either confidentiality or integrity is selected, a session key must be established. A random master key of 256 bits is generated. The key is split in half to create two 128 bit subkeys. One key is used for symmetric encryption and the other key is used for message authentication codes.

The client will first generate the master key. Next, the master key is encrypted with the server's public key and sent to the server. This encryption protects the session keys during transit, since only the server can decrypt them.

Confidentiality

Messages between the client and server are encrypted using the AES/CBC/PKCS5Padding cipher. The encryption includes a random 16 byte initialization vector. The initialization vector is send along with the encrypted message, since it is required for encryption. The encryption/decryption process uses the established session keys.

Integrity

If message integrity is required, messages between the client and server also include a message authentication code (MAC). The encrypted/plaintext messages are hashed using the HMACSHA256 algorithm and encrypted with the second session key. The MAC is appended to the encrypted message. Once the message is received, the receiver will compute a MAC from the message data. If the computed MAC matches the MAC appended to the message, the message is authentic and verified.

Authentication

Two different authentication mechanisms are implemented.

Password-Based Auth

If authentication is required, the server and client will prompt for a username and password. The user must enter proper credentials on the server and client to continue program execution. The user credentials are stored in UserDB as a hash of the password and a random salt. The salt is stored in plaintext. The input credentials are hashed with the random salt, and the hash is compared with the hash from the server. If the hashes match, authentication succeeds.

Certificate-Based Auth

If authentication is required, the client and server will exchange signed certificates from their KeyStores. Mutual authentication succeeds if both the client and server verify each other's signed certificates.