Developed by: *Ben Wolfe (V00205547) and Daniel Olaya (V00855054)*

Course: *SENG360 - Security Engineering*

Date: *November, 2017*

Assignment 3: *Chat program that allows to users to communicate using different security parameters, encryption, integrity (digital signatures) and mutual authentication.*

Required files

The following files are required to run the program:

- Client.java
- Server.java
- AsymmetricCryptography.java
- ClientSignature.java
- ServerSignature.java
- Servercred/credentials.txt
- Servercred/clientRSApublicKey
- Servercred/serverRSAprivateKey
- Servercred/serverRSApublicKey
- Servercred/serverPublicKeySignature
- Clientcred/credentials.txt
- Clientcred/serverRSApublicKey
- Clientcred/clientRSAprivateKey
- Clientcred/clientRSApublicKey
- Clientcred/clientPublicKeySignature
- bouncyCastle.jar
- commons-codec-1.7.jar

Assumptions

- Client and server have shared their RSA public keys that will be used to encryption, this can be done posting the information online since it does not represent a security risk for the users.

Compiling program

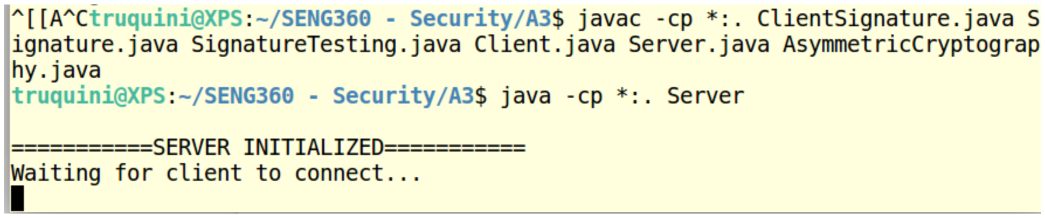
Execute following command

```
javac -cp *:. ClientSignature.java ServerSignature.java
SignatureTesting.java Client.java Server.java
AsymmetricCryptography.java
```

Running the program

Execute following command on a terminal window

```
java -cp *:. Server
```

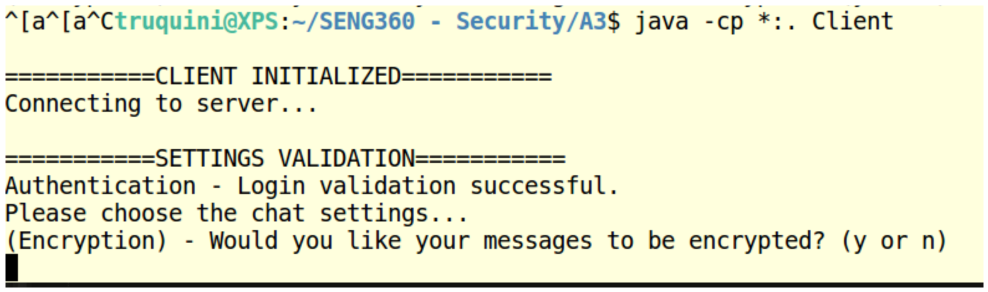


```
^[[A^Ctruquini@XPS:~/SENG360 - Security/A3$ javac -cp *:. ClientSignature.java S
ignature.java SignatureTesting.java Client.java Server.java AsymmetricCryptograp
hy.java
truquini@XPS:~/SENG360 - Security/A3$ java -cp *:. Server

===========SERVER INITIALIZED===========
Waiting for client to connect...
```

Execute following command on a different terminal window

```
java -cp *:. Client
```

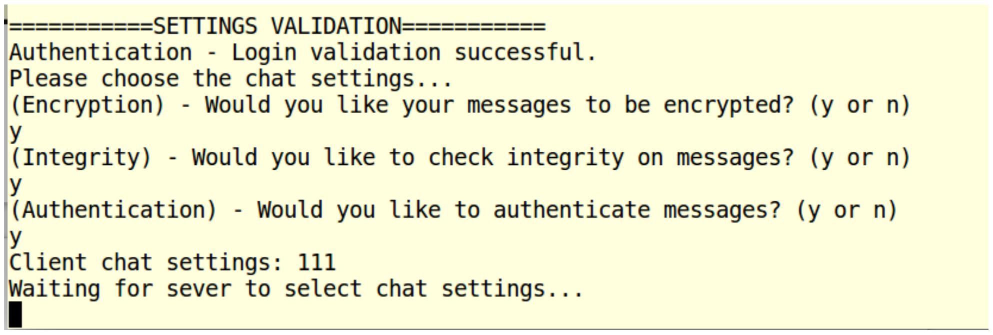


```
^[a^[a^Ctruquini@XPS:~/SENG360 - Security/A3$ java -cp *:. Client

===========CLIENT INITIALIZED===========
Connecting to server...

===========SETTINGS VALIDATION===========
Authentication - Login validation successful.
Please choose the chat settings...
(Encryption) - Would you like your messages to be encrypted? (y or n)
```

The program will request the client to enter chat settings, use ‘y’ or ‘n’ to select settings. The program will send selected settings and will request the server to enter its desired settings. If both settings are not equal, the connection will restart until both Server and Client have selected the same settings.



```
===========SETTINGS VALIDATION===========
Authentication - Login validation successful.
Please choose the chat settings...
(Encryption) - Would you like your messages to be encrypted? (y or n)
y
(Integrity) - Would you like to check integrity on messages? (y or n)
y
(Authentication) - Would you like to authenticate messages? (y or n)
y
Client chat settings: 111
Waiting for sever to select chat settings...
```

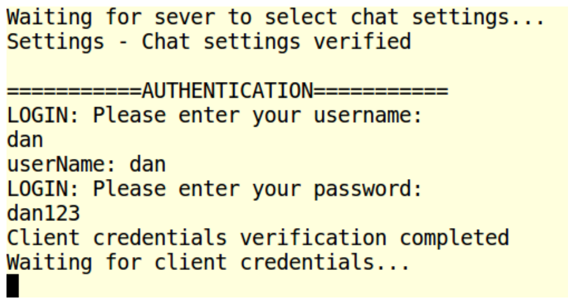
Authentication

If authentication is selected, both the client and server will authenticate to each other. The program hashes the password given by Client and sends it to the server. The server will check the ‘username’ is exists in its login database located in `/Servercred/credentials.txt` if the login is found, it will proceed to take the salt and hash it with the hash(pass) given. If the result of hash(salt+hash(pass)) matches with the saved entry, the login is valid. If the client login is valid, it will proceed to complete the same process with the server.

WARNING: A list of the plain-text credentials and salt can be found in

`/Servercred/credentials-plain.txt` this file only exists for assignment grading purposes and should never exist in a production environment.

Proceed to login in Client with username: ‘dan’ and password ‘dan123’.



```
Waiting for sever to select chat settings...
Settings - Chat settings verified

===========AUTHENTICATION===========
LOGIN: Please enter your username:
dan
userName: dan
LOGIN: Please enter your password:
dan123
Client credentials verification completed
Waiting for client credentials...
```

Enter following credentials in Server, username: ‘ben’ and password ‘ben123’.

If wrong password is given by Server or Client, the connection will be restarted for both users.

Integrity

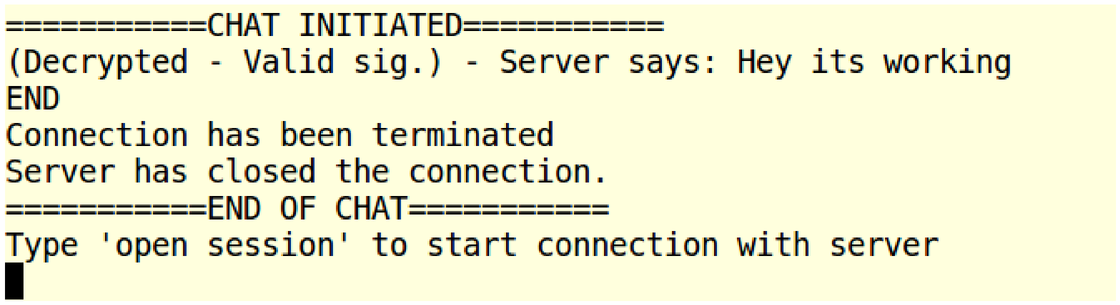
If integrity option is selected, the program will initialize keys that will be used to sign a message (Private key) and validate a message (Public key) using helper class `ServerSignature.java` and `ClientSignature.java`

Encryption

If the encryption option is selected, the previously generated keys using the helper class `GenerateKeys.java` will be initialized using helper class `AsymmetricCryptography.java`

Encryption

Once all key generation and initialization has been completed, listening and receiving threads will start. Use ‘`END`’ to finish the conversation. After the conversation is closed, the Server will restart the program and wait for a new connection to be initiated. In the case of the client, it will be able to restart the connection any time by using the command ‘`open session`’.



```
===========CHAT INITIATED===========
(Decrypted - Valid sig.) - Server says: Hey its working
END
Connection has been terminated
Server has closed the connection.
===========END OF CHAT===========
Type 'open session' to start connection with server
█
```