SENG 360
Assignment 360
Due: November 9,2017
Russell Snelgrove
Nik Rados

Classes/Functionality:

Client:

1.Global variables (Type):

1. ALGO (String): This is the variable that we use to determine algorithm is used in the generation of keys for encryption and decryption.
2. keyValue (byte): The values that we use for the encryption and decryption.
3. check_file(string): name of the file that we use when checking integrity.

2. Methods (parameters):

1. encrypt(String Data): This is the function that is called when Confidentiality is selected to encrypt the message prior to being sent, it takes in a string that is unencrypted and encrypts it before sending the message to the server. This function uses AES encryption.
2. decrypt(String encryptedData): This is the function that is called when Confidentiality is selected to decrypt the message prior to being printed to the command prompt, it takes in a string that is encrypted and decrypts it. This function uses AES decryption.
3. generateKey(): Generates the key used in encryption and decryption using the ALGO and keyvalue global variables.
4. getCIA(): This function runs the user through a text interface that decides the Security properties that the user wishes for.
5. bit_shit(String change): Here we bit shift a string to be written into a file where we use the written message as a check for the message that is sent.
6. write_text(String write_string): This is where we write the bit shifted value into a file and overwrite all old content, this ensures that we are writing the “freshest” content.
7. compare_text(String to_check): This is the method that we use to compare the string sent against the string that is bit shifted. In this method we call the bit shift to bit shift the string that was send through the message and then compare it to the text in the file. This insures that even if someone intercepted the message they would need to bit shift before confirming its integrity.
8. main(): The client starts in the main by establishing connection with the server. Then proceeding to ask the user for what of the 3 security properties they wish to implement. After the user selects the properties they are then sent off to the server and checked by the server. When the server deems that the properties are the same or different a response will be recieve indicating which of the two cases

took place. At this point the client enters its while loop where the remainder of the conversation takes place. If Confidentiality was selected we encrypt every message before sending it and decrypt every message before printing it to the command prompt. If Authentication is selected the client will be prompted by the server to enter a username and password, the default username is seng360 and the password is assignment3. After logging in if required the client will send the first message the server may respond and the conversation will take place by a rotation of messaging, one to one ratio. If either of the parties put into the command prompt “!quit” the program will terminate.

Server:

1. Global variables (Type):
    1. socket (Socket): This is the socket that the server runs on and accepts a connection from.
    2. Authentication (int): This is the Authentication, that I may use in the program to determine if Authentication is used, most likely it will be changed to a boolean like we did with Integrity and Confidentiality.
    3. Integrity (boolean): This is the boolean that represents if the Integrity security property was selected.
    4. Confidentiality (boolean): This is the boolean that represents if the Confidentiality security property was selected.
    5. Command_total (int): This is the added up total of the security properties so we can think about the selection as a binary. The first bit being Confidentiality followed by Integrity and Authentication.
    6. Running (boolean): This is the boolean I use for the main when the program is running, when this value is turned false the program will terminate after the encounter is finished with the client.
    7. filename (String): The filename of the file that we store the user and password of intended users.
    8. DEFAULT_USER (String): This is set to seng360 which is the username we have selected to work.
    9. DEFAULT_PASS (String): This set to assignment3 which is the password that we have selected to work.
    10. ALGO (String): This is the variable that we use to determine algorithm is used in the generation of keys for encryption and decryption.
    11. keyValue (byte): The values that we use for the encryption and decryption.
    12. check_file(string): name of the file that we use when checking integrity.

2. Methods (parameters):
    1. selected(): This is a method created to remind the user what security properties have been selected and sets the boolean security properties to true or false.

2. getCIA(): Gets the user to input the desired security properties that they wish to incorporate.
3. check_user(String name, String pword): Checks to see if the username and password inputted from the client matches the username and password in the “data base”. If both properties do not match up the return value is false otherwise true is returned.
4. generateUserPass(String username, String password): This function takes in the username and password that are to be used in the program, hashes them then stores them into our “data base”
5. String hashText(String unhashed): This function will take in the desired unhashed string, hash the string and return the hashed string.
6. writeToFile(String username, String password): This is the function that is called to write the hashed username and password to the file that was defined.
7. encrypt(String Data): This is the function that is called when Confidentiality is selected to encrypt the message prior to being sent, it takes in a string that is unencrypted and encrypts it before sending the message to the client. This function uses AES encryption.
8. decrypt(String encryptedData): This is the function that is called when Confidentiality is selected to decrypt the message prior to being printed to the command prompt, it takes in a string that is encrypted and decrypts it. This function uses AES decryption.
9. generateKey(): Generates the key used in encryption and decryption using the ALGO and keyvalue global variables.
10. bit_shit(String change): Here we bit shift a string to be written into a file where we use the written message as a check for the message that is sent.
11. write_text(String write_string): This is where we write the bit shifted value into a file and overwrite all old content, this ensures that we are writing the “freshest” content.
12. compare_text(String to_check): This is the method that we use to compare the string sent against the string that is bit shifted. In this method we call the bit shift to bit shift the string that was send through the message and then compare it to the text in the file. This insures that even if someone intercepted the message they would need to bit shift before confirming its integrity.
13. main(): This is the meat of this class. In this main method we start by making a connection the port that can be changed but since we managed to get 7802 to work we have stuck with that. We start up the socket then go to call getCIA(), then call selected() letting the user know what properties he/she selected. We then enter the while loop that the server will sit in until the program is ended. We begin exchanging messages with the client at this point going through the formalities of first establishing if the selected properties are the same. After finding if the selected properties are the same we then proceed using the properties that we selected. If Confidentiality was selected we encrypt every message before sending it and decrypt every message before printing it to the

command prompt. If Authentication is selected the client will be prompted by the server to enter a username and password, the default username is seng360 and the password is assignment3. After the conversation has begun the client will send the first message the server may respond and the conversation will take place by a rotation of messaging, one to one ratio. If either of the parties put into the command prompt “!quit” the program will terminate. If the server did not terminate the session it will remain running until a new client connects and the process of communication may repeat.

Program Overview:

The program is primarily based around the exchange of messages between two parties. These two parties are the Server and Client programs that have been written in their respective .java files. The programs run together and are based out of a socket connection using local hosting. The programs allow the users to select between what security properties that they wish to implement Confidentiality, Integrity, and Authentication. If Confidentiality is selected each message will be encrypted before being sent to the opposing party and the receiving party will then decrypt the message using AES encryption/decryption. If Integrity is selected the message prior to being sent will be attached with a digital signature, and using ban logic we just assume that if the client or server believe the signature of the other party they then also believe in the message being from the other party. If the Authentication property is selected the server will prompt the client for a username and password and is preset and hashed to a table that the server takes care of. The client the enters its credentials and is either granted access or told that there is either an error with the password or username. Once the conversation is underway the client sends the first message and the server sends the second message. This conversation will be a one message rotation conversation. This meaning that one message can be sent per a party until they must wait for a reply from the other party. This is similar to how the military uses the keyword over when done their side of a conversation and is awaiting a response from the other end. To end the conversation either party may enter “!quit” and the program will end. Throughout the program if integrity is selected both of the programs write the message that they are sending into a file and it is bit shifted in the file. The receiving party selects the received message and bit shifts it and compares the string. If the string is verified it is then consisting of integrity.

To Run:

In order to run these programs successfully the server must first be ran by compiling it using <javac Server.java> then ran using <java Server>. After being run the Server must run through the security questions requesting what properties the server is wanting to implement. Then it will say that the server is ready for a client to connect. At which point we are looking to booting up the client. We do this by compiling using <javac Client.java> and the running it using

<java Client>. After we then proceed to completing the same question asked to the server. If the answers do not match the client and server will be notified. If the answers match we then proceed to have a conversation. The client and server may communicate freely with a one to one message useage. The client may send the first message followed by the server's message and it will rotate until either of the two parties input “!quit” and then the program will quit.