

Two-way handshake in TCP.

İbrahim Burak GÜLER

# 1.Abstract

In this project, using, two way handshake, reliability is supported on TCP protocol. Therefore, client party and server party are connected if and only if they are verified by two way handshake. If one party does not have this protocol, other party refuse the connection because of no validation. Also, reliability of the protocol has been tested by one application which is named “Wireshark”. Moreover, normal TCP data transfer uses to compare with this project. Addition to this, screen shots are provided according to the result has been captured along the project. In the conclusion of project, according to the experiment results, some useful assumptions will be given.

## 2.Implementation Details

- sslClient.java

In this class, firstly we create the sslSocket in order to connect the other party with this sslSocket. If ip, port and ssl key is correct the connection is established and the client displays other party's public key thanks to display\_public\_key() method. After that, Client gets the file from server side and save it specific directory. Finally, the connection has terminated after the file transportation is done. If key is not correct key or ip and port numbers are not correct program throw errors.

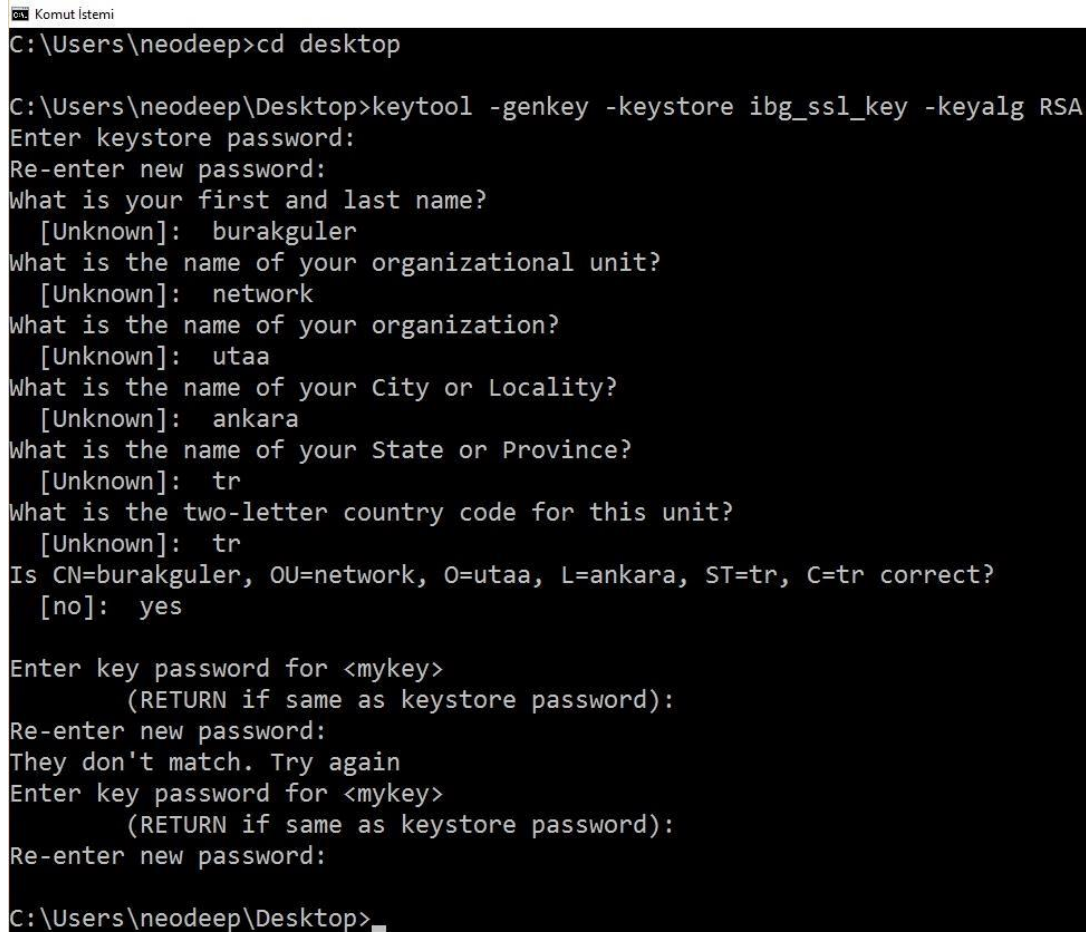
**-sslServer.java**

Firstly, In this class, SSLSockets are used to provide the security again. After establish the connection, like client side, the server displays other party's public key thanks to display\_public\_key(). After that, server send a file to the client using

this port and shut the connection. If key is not correct key or ip and port numbers are not correct program throw errors.

#### -Certificate ibg\_ssl\_key

We have been created the certificate file with the following command in the command line: “keytool -genkey -keystore mySrvKeystore -keyalg RSA”.(figure 1)



```
Komut İstemi
C:\Users\neodeep>cd desktop

C:\Users\neodeep\Desktop>keytool -genkey -keystore ibg_ssl_key -keyalg RSA
Enter keystore password:
Re-enter new password:
What is your first and last name?
  [Unknown]: burakguler
What is the name of your organizational unit?
  [Unknown]: network
What is the name of your organization?
  [Unknown]: utaa
What is the name of your City or Locality?
  [Unknown]: ankara
What is the name of your State or Province?
  [Unknown]: tr
What is the two-letter country code for this unit?
  [Unknown]: tr
Is CN=burakguler, OU=network, O=utaa, L=ankara, ST=tr, C=tr correct?
  [no]: yes

Enter key password for <mykey>
  (RETURN if same as keystore password):
Re-enter new password:
They don't match. Try again
Enter key password for <mykey>
  (RETURN if same as keystore password):
Re-enter new password:

C:\Users\neodeep\Desktop>
```

*Figure 1:Creating the SSL Certification File*

### 3.Experimental Result

In this experiment First of all, normal TCP without security protocols, is used for sniffing the messages on the defined port by using “Wireshark” application. For this purpose, messegases traffic between two computers has been captured and the comparison which defines the differences between normal TCP and Reliable Protocol. The port that has the messages between two computers have been listened and the messegas easily could be read.(Figure 2)

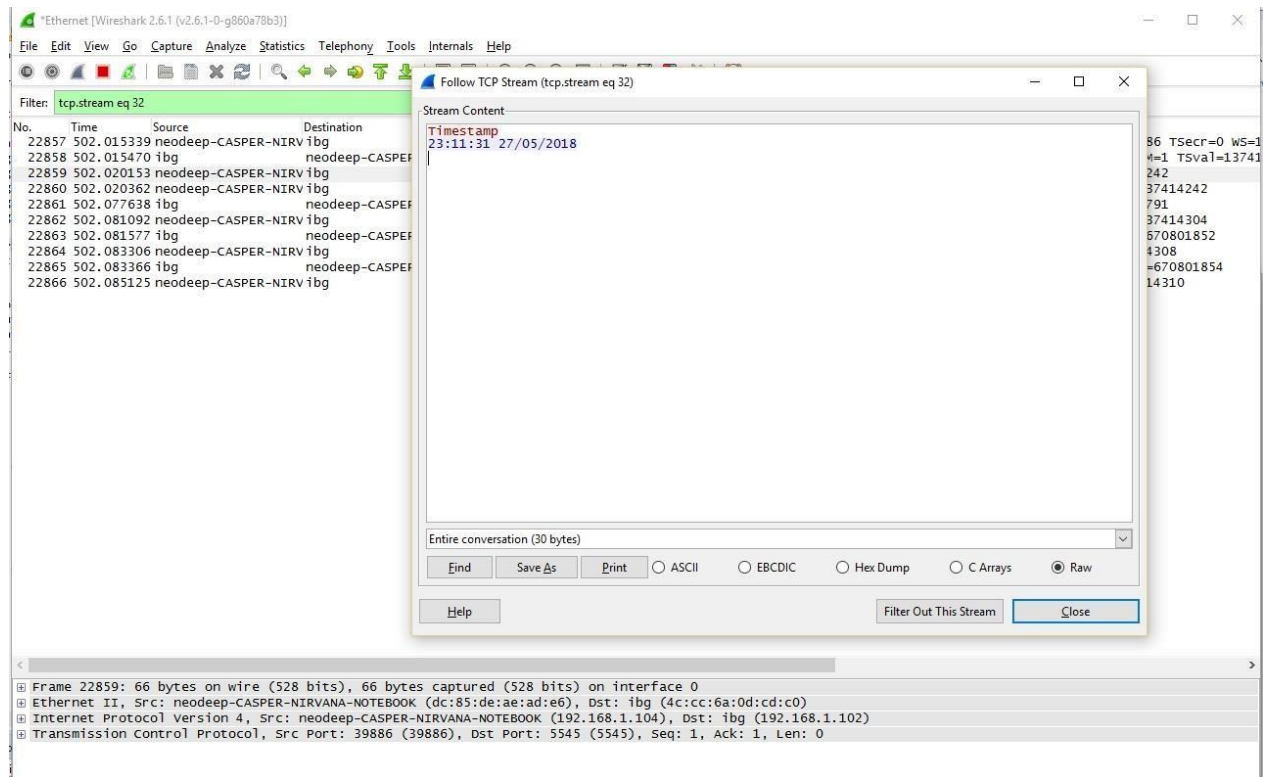


Figure 2:Standart TCP Protocol Sniffing port

When we used Reliable TCP protocol, we observed that we can not reach the message because of security protocols and certification. All data were encrypted. (Figure 3)

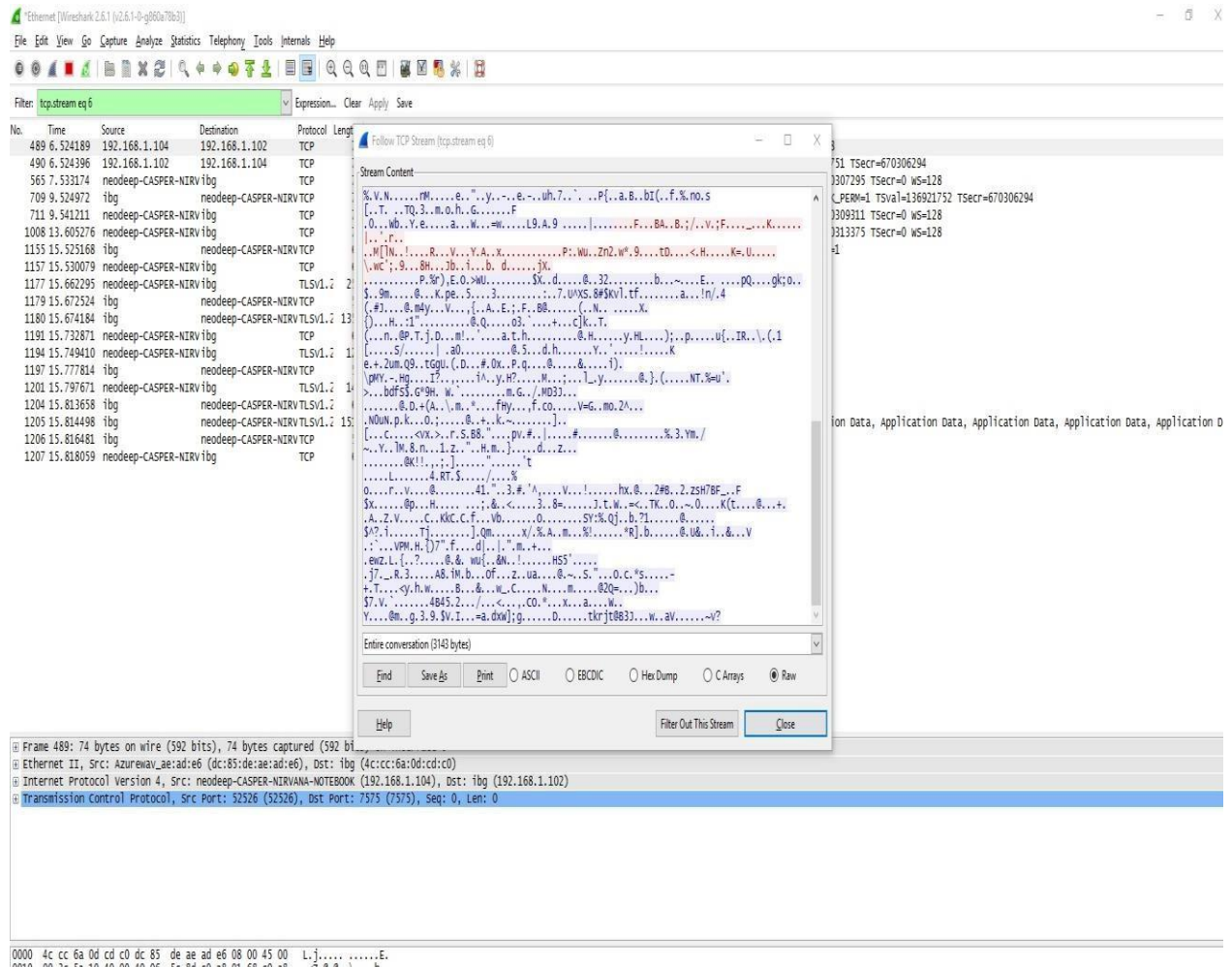


Figure 3:Reliable TCP Protocol Sniffing port.

## 4. Screen Shots

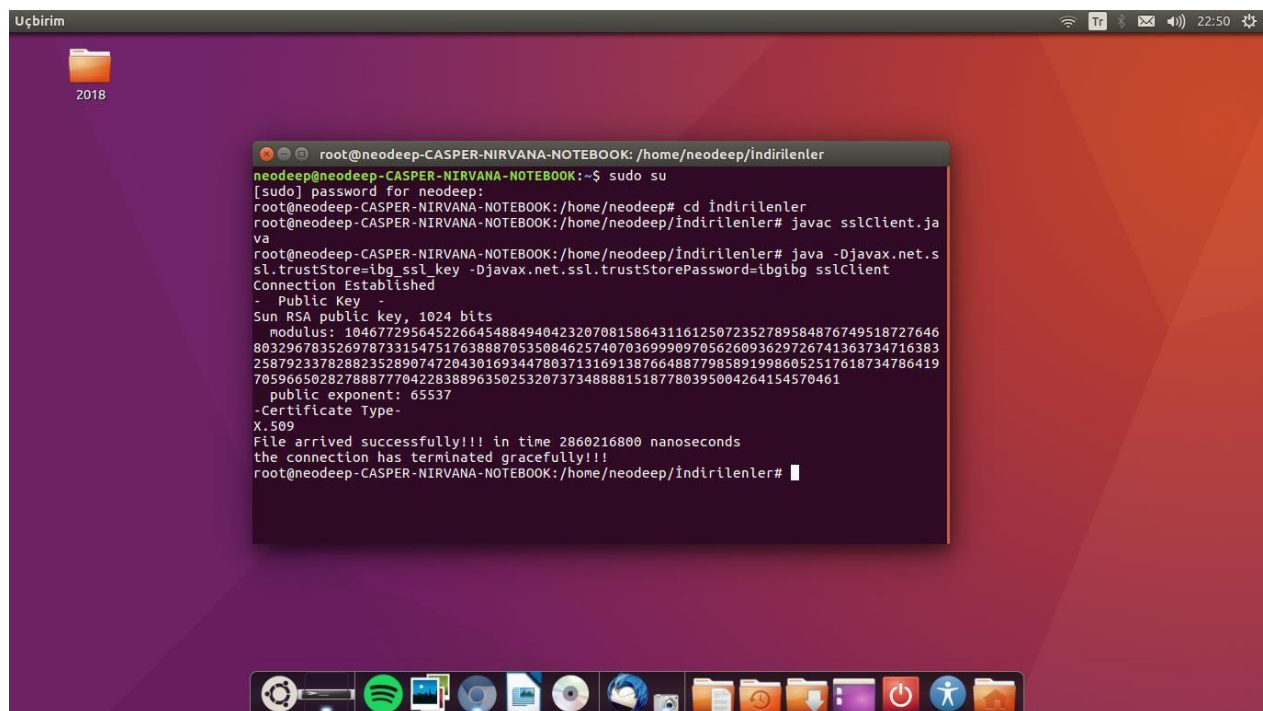
### 4.1. Running

In this project, firstly ssl certificate was created using command line with \$keytool method. “keytool -genkey -keystore ibg\_ssl\_key -keyalg RSA”. Later that this ssl certificate is added to the project for both parties(Clientside and Serverside).

There are some steps to be able to run the project on the command line. Firstly, using “javac” key SecureServer class must be compile and run, later clientside “SecureClient.java” must be compile and run. After compiling, to be able to run the both sides following commands are needed to use

Java -Djavax.net.ssl.keyStore= ibg\_ssl\_key -Djavax.net.ssl.keyStorePassword=ibgibg sslServer  
(Figure 4)

Java -Djavax.net.ssl.trustStore= ibg\_ssl\_key -Djavax.net.ssl.trustStorePassword=ibgibg sslClient  
(Figure 5)

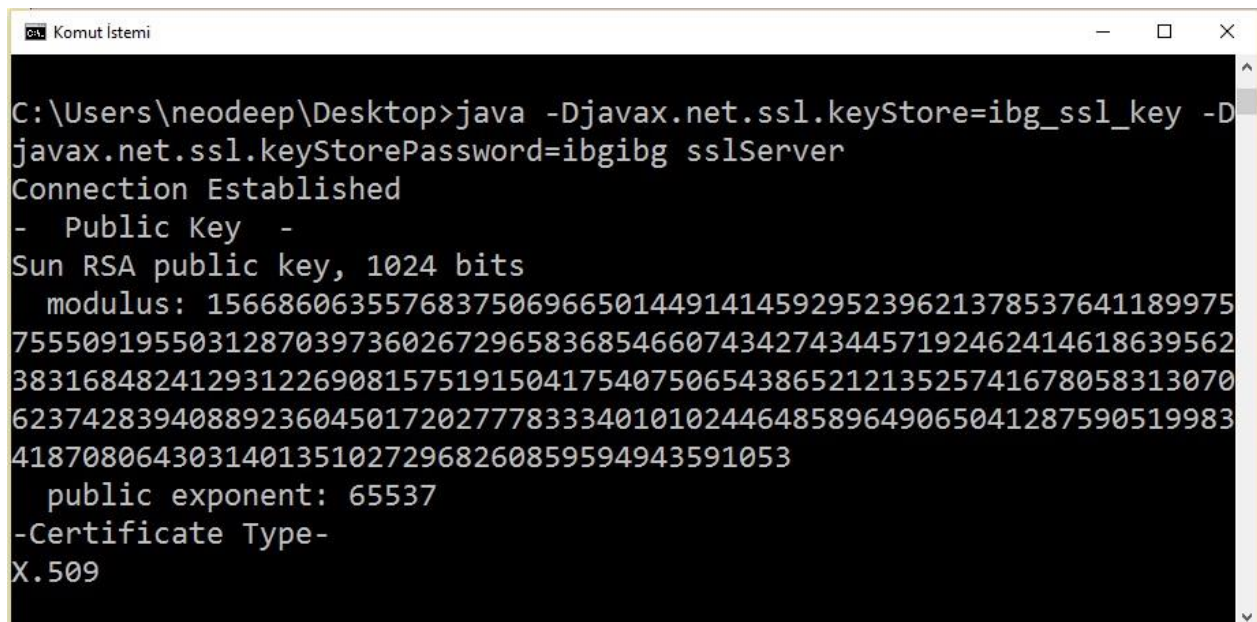


```
Uçbirim
2018

root@neodeep-CASPER-NIRVANA-NOTEBOOK: /home/neodeep/indirilenler
neodeep@neodeep-CASPER-NIRVANA-NOTEBOOK:~$ sudo su
[sudo] password for neodeep:
root@neodeep-CASPER-NIRVANA-NOTEBOOK:/home/neodeep# cd indirilenler
root@neodeep-CASPER-NIRVANA-NOTEBOOK:/home/neodeep/indirilenler# javac sslClient.java
root@neodeep-CASPER-NIRVANA-NOTEBOOK:/home/neodeep/indirilenler# java -Djavax.net.ssl.trustStore=ibg_ssl_key -Djavax.net.ssl.trustStorePassword=ibgibg sslClient
Connection Established
- Public Key -
Sun RSA public key, 1024 bits
modulus: 104677295645226645488494042320708158643116125072352789584876749518727646
80329678352697873315475176388870535084625740703699909705626093629726741363734716383
25879233782882352890747204301693447803713169138766488779858919986052517618734786419
7059665028278887704228388963502532073734888815187780395004264154570461
public exponent: 65537
-Certificate Type-
X.509
File arrived successfully!!! in time 2860216800 nanoseconds
the connection has terminated gracefully!!!
root@neodeep-CASPER-NIRVANA-NOTEBOOK:/home/neodeep/indirilenler#
```

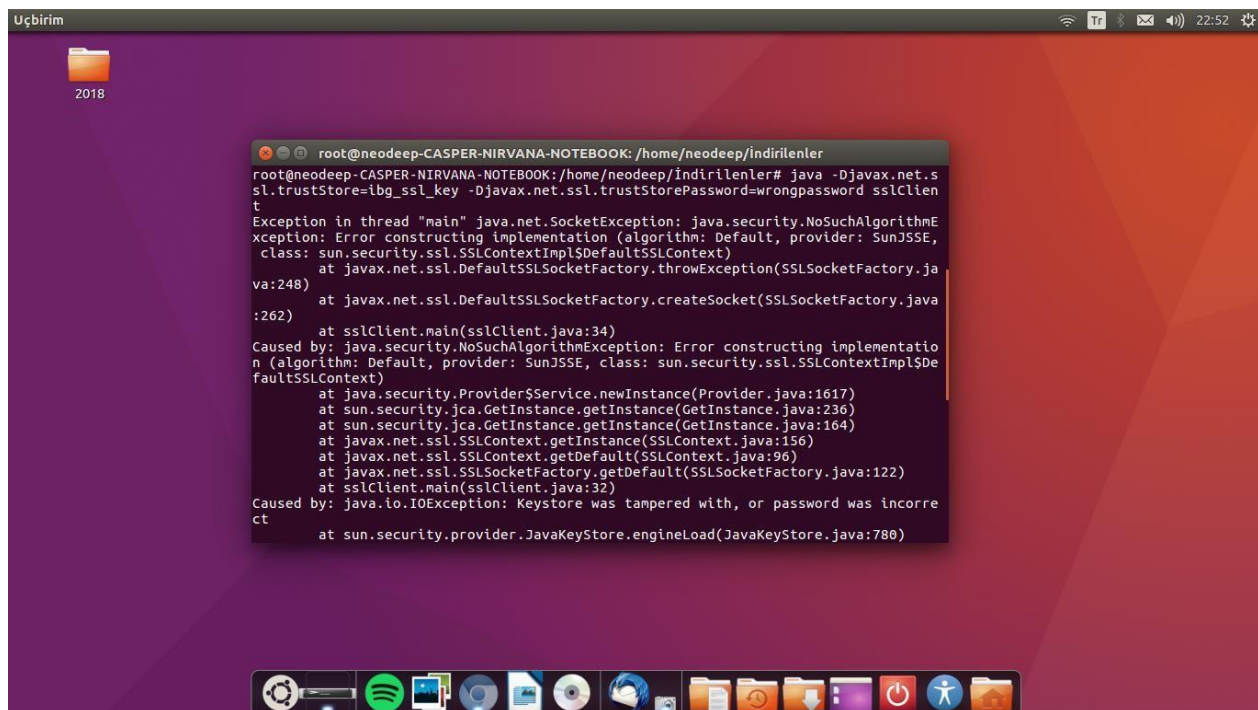
Figure 4:Reliable TCP Protocol client side.





```
C:\Users\neodeep\Desktop>java -Djavax.net.ssl.keyStore=ibg_ssl_key -Djavax.net.ssl.keyStorePassword=ibgibg sslServer
Connection Established
- Public Key -
Sun RSA public key, 1024 bits
  modulus: 1566860635576837506966501449141459295239621378537641189975
755509195503128703973602672965836854660743427434457192462414618639562
383168482412931226908157519150417540750654386521213525741678058313070
623742839408892360450172027778333401010244648589649065041287590519983
41870806430314013510272968260859594943591053
  public exponent: 65537
-Certificate Type-
X.509
```

Figure 5:Reliable TCP Protocol server side.



```
root@neodeep-CASPER-NIRVANA-NOTEBOOK: /home/neodeep/indirilenler# java -Djavax.net.s
ssl.trustStore=ibg_ssl_key -Djavax.net.ssl.trustStorePassword=wrongpassword sslClien
t
Exception in thread "main" java.net.SocketException: java.security.NoSuchAlgorithME
xception: Error constructing implementation (algorithm: Default, provider: SunJSSE,
class: sun.security.ssl.SSLContextImpl$DefaultSSLContext)
at javax.net.ssl.DefaultSSLContextFactory.throwException(SSLContextFactory.java
:248)
at javax.net.ssl.DefaultSSLContextFactory.createSocket(SSLContextFactory.java
:262)
at sslClient.main(sslClient.java:34)
Caused by: java.security.NoSuchAlgorithmException: Error constructing implementatio
n (algorithm: Default, provider: SunJSSE, class: sun.security.ssl.SSLContextImpl$De
faultSSLContext)
at java.security.Provider$Service.newInstance(Provider.java:1617)
at sun.security.jca.GetInstance.getInstance(GetInstance.java:236)
at sun.security.jca.GetInstance.getInstance(GetInstance.java:164)
at javax.net.ssl.SSLContext.getInstance(SSLContext.java:156)
at javax.net.ssl.SSLContext.getDefault(SSLContext.java:96)
at javax.net.ssl.SSLContextFactory.getDefault(SSLContextFactory.java:122)
at sslClient.main(sslClient.java:32)
Caused by: java.io.IOException: Keystore was tampered with, or password was incorre
ct
at sun.security.provider.JavaKeyStore.engineLoad(JavaKeyStore.java:780)
```

Figure 6:Reliable TCP Protocol wrong key error.

## 5.Conclusion

To have a look at what we have done in this project and results that we have gathered on our experimental works, there are some major points to talk about. First of all, we implemented a reliable TCP protocol using java programming language. To achieve this, we have used ssl certificate and two way handshake which covers the both client and server side. To sniff the packets between two computers' stream, we have used “wireshark” application and help of wireshark, we checked our Reliable TCP works correctly and in a secure way. Then, we observed Reliable TCP makes it very secure to connect client to server and sending the data from server to the client. Therefore, we presented the differences between standart TCP and reliable TCP protocol which we implemented. So that, we figured it out that while packets are visible sent by standart protocol, packets sent by reliable TCP are not visible and those data are secure.