CYBR372 Assignment 2

Part 2

```
C:\Program Files\Java\jre1.8.0_201\bin>keytool -list -keystore cybr372.jks
Enter keystore password:
Keystore type: jks
Keystore provider: SUN

Your keystore contains 2 entries

client, 11/10/2020, PrivateKeyEntry,
Certificate fingerprint (SHA1): A2:17:74:D0:A4:73:51:AE:9A:DD:53:8B:CE:F2:DD:4D:F2:3F:4E:E9
server, 11/10/2020, PrivateKeyEntry,
Certificate fingerprint (SHA1): 14:1D:D7:EA:23:EE:7C:23:57:69:FC:0C:75:A1:21:C7:DF:F1:44:B4
```

A key tool has been created that has two entries in it, client and server that contain a certificate and a private key for both the client and the server. The code now uses the information in the key store to encrypt and decrypt the messages rather than copying and pasting them in the terminal.

Server:

```
$ java EchoServer "C:\Program Files\Java\jre1.8.0_201\bin\cybr372.jks" badpassword serverpassword
Stored keys at C:\Program Files\Java\jre1.8.0_201\bin\cybr372.jks
Waiting to complete Exchange
Server received cleartext 12345678
Signature Valid
Server sending ciphertext MfaIQREqUdhSJVOLtLauQCa4/E3gahOKaiD8r/2lbiVIKBTFp8VFxqfbPn3B5byuTrG9pCJVfI82q6xOihI93vYgFx485sJgirY5HWgdkcF1
fNuYWChP8wpsRgte6a6RyF+OVVhPopTPdViGturJ+Xtl2w2cXOICXy9q5Aa6JZyu2GTtgt+BnK7vqz+UBqcHEQEVSOy4O+u2BOULnkb5ItWXwsPRUg4NHvQizOO/4x6BZJNBTd
gCtPrGyzUZ3pOegEo5iYcjIJJZEXXr86i2pXZhE5G3pGK/GyEQXrxQR7IMOOokJyGYwWDQhraiLXQORqQV2WSRfzQVGUrCNWDDCw==
```

The server takes in three arguments, the location of the key store, the key store password and server key password. The program then uses the key store to get the public key for the client and uses the password to get the private key for the server. It then follows the same encryption and decryption process as part 1.

Client:

```
$ java EchoClient "C:\Program Files\Java\jre1.8.0_201\bin\cybr372.jks" badpassword clientpassword
Stored keys at C:\Program Files\Java\jre1.8.0_201\bin\cybr372.jks
Keys exchanged
Client sending cleartext 12345678
Client sending ciphertext XUA6RpVQsgpWeaD68oSg6OF4928On1zcAvmT15aitMALU1AqhT7hOAbp757rOwDC/C6Qx12OnaMUrtnuAzp4/yBF7MXrOtM3POmdO6id6x8x
ASFxMFx/ObvQAMylJsT9uukw/Hn2HrahJV968TDtCcwl9DejlCj6trfavZfE3GherwN+7V7ZD11R8hqSl5UgfZ2HIhUYhqumFQUO42/KYgHudGaFBLFDqEd1IQqGdYfI5yeWsJ
CQvZhgAapwn8/Zv+xoEIOVn8uQTs3G+jbl4m/6zkiY4CMom1LqImjuD293fXHtfrOe9bYRpOsng3Xy1dBefeUOznA88Ulx9qRhEw==
Client received cleartext 12345678
Signature Valid
```

The client takes in three arguments, the location of the key store, the key store password and client key password. The program then uses the key store to get the public key for the server and uses the password to get the private key for the client. It then follows the same encryption and decryption as part 1.

Running Part 2:

Client Arguments: (Key Store location) (Key Store Password) (Client password)

Server Arguments: (Key Store location) (Key Store Password) (Server password)

Key Store Password: badpassword

Client Password: clientpassword

Server Password: serverpassword

Design Decisions:

Part 2 mostly improves part 1. It removes the need for the public keys to be pasted around and uses the key store instead. The code reads from the key store and then continues to encrypt and decrypt like Part 1.

I have chosen to only allow the programs to receive the password for the key store that they require. For example, the client only receives the key store password and the client password. This means that the client can't access the server's private key which is something that it shouldn't have access to as it is called a private key for a reason. This is part of the reason that everything has a different password because it means that I can ensure that the client can't access the server's key and vice versa.