# Programming Assignment 2 Checklist

**We only have 15 minutes** for the demo per pair. In this 15 minutes, we need to do all these:

1. Demo your program with various file sizes, for both CP1 and CP2 (about 3 files per CP) -- small files <1KB, medium files ~1-100MB, large files >=100MB
    a. These files are meant to be downloaded from edimension on the day of your checkoff,
    b. But you need to prepare several files like these also in case edimension download doesn't work
    c. Upload multiple files in 1 connection through one of *two ways (basically you can't recompile to upload multiple files!):*
        i. Give filename from arguments
        ii. Give commands to server back and forth like the demo video
    **d. Demonstrate that your code terminate properly (no crashing, no error messages at the console)**
2. Answer questions from us:
    a. Conceptual question
    b. Pertaining to your code
    c. Any bonus part that you have done

**Note which line numbers on your code where it does the following (variable names may vary):**

1. On the client side CP1:
    1. Get server.crt from server: **fromServer.readFully(certificate, 0, numBytes);**
    2. Verify (and decrypt) the server.crt using CA cert (cacse.crt): **ServerCert.verify(CaKey);**
    3. Extract server's public key from the certificate: **Serverpublickey = ServerCert.getPublicKey();**
    4. Encrypt file chunks with server's public key: **byte[] cipherTextLong = cipherServer.doFinal(last);**

2. On the client side CP2:
    1. Generate symmetric key: **KeyGenerator keyGen = KeyGenerator.getInstance("AES");**
    2. Send symmetric key to server (encrypted with server's public key): **byte[] cipherTextLong = cipherServer.doFinal(last); toServer.write(encryptSymmetricKey);**
    3. Encrypt file chunk with symmetric key: **byte[] cipherTextLong = aesCipher.doFinal(last);**

3. On the server side CP1:
    1. Sending of server certificate to the client: **toClient.write(certByte);**

2. Encrypt nonce with private key: **byte[] encryptedNonce = rsaCipherEncrypt.doFinal(nonce);**
3. Decrypt file chunks with private key: **byte[] aesKeybytesDecrypted = rsaCipherDecrypt.doFinal(aesKeybytesEncrypted);**

4. On the server side CP2:
    1. Decrypt symmetric key with private key: **byte[] aesKeybytesDecryptedKey = rsaCipherDecrypt.doFinal(aesKeybytesEncryptedKey);**
    2. Decrypt file chunks with symmetric key: **decryptedblock = aesCipherDecryptor.doFinal(encryptedblock);**

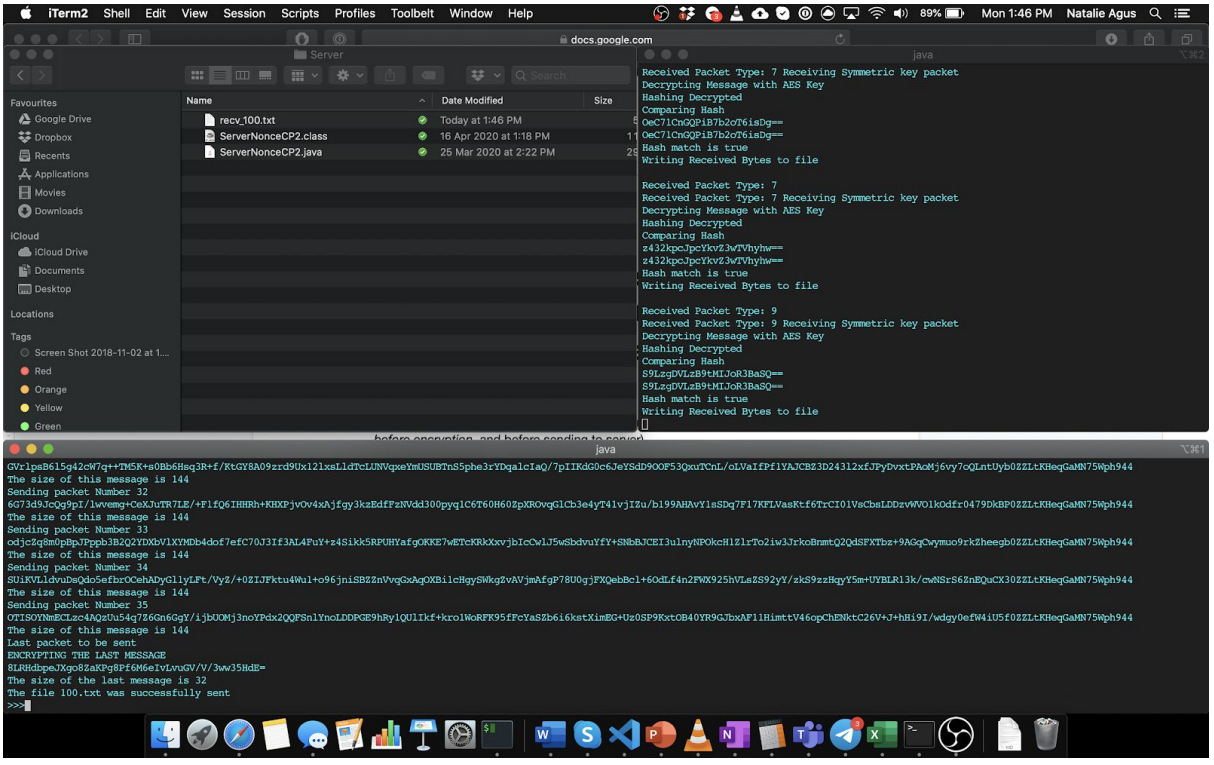
**Also, prepare these lines. Comment them out at first for the demo. Only uncomment them and recompile when *we ask you to.***

1. Print unencrypted bytes on client side in CP1 and CP2 (these are the file chunks *before encryption,* and before sending to server)
2. Print encrypted bytes on client side in CP1 and CP2 (these are the file chunks *after encryption,* and before sending to server)
3. Print received encrypted bytes on server side in CP1 and CP2 (these are the from client file chunks *before decryption*)
4. Print unencrypted bytes on server side in CP1 and CP2 (these are the file chunks from client *after decryption*)

You may use the Base64Encoder we used in Lab 6 to print Byte Arrays, or simply just do System.out.println(new String(byteArrayBuffer));

During the demo, arrange your screen to have the following windows:

1. Opened working directory of the server
2. Terminal for the server
3. Terminal for the client




We want to see that new files pop up in the working directory of the server when you send them.

**Be punctual in your allocated slot. If you are late, we wouldn’t replace your lost time. Go to the channel that you have signed up, and join the meeting at your exact time slot. DO NOT join early.**