COMPSCI 711 A1 Report

(a) I have implemented option 2.

(b) I have not had the opportunity to test my programs on any lab machines as I haven't been in Auckland for the past month.

(c) Before testing the program, please read the README.md file for more in-depth instructions. There are four main files that need to be opened:

   a. The first file to open is the executable file "Server 6.0.exe" in the folder "Build - Server".
   b. The second file to open is the executable file "Cache 6.0.exe" in the folder "Build - Cache".
   c. The third file to open is the executable file "Client 6.0.exe" in the folder "Build - Client".
   d. The final file is a batch file called "Start.bat" inside the main folder.

   Once these files are opened you are ready to test. You can put your test images into a folder called "Images". You can find this folder in the directory called "…/Build – Server/wwwroot/Images". Once these files are in this folder, you can head over to the GUI for the cache and click the "Update Server/Cache" button. This will update the server and cache. You should be able to see the available files in the cache. To see the files in the client GUI, you will need to refresh the webpage. After following these steps, you should be able to download the images to the client GUI and inspect the downloaded data in the cache GUI.

(d) When transferring the files, their data is converted into a base64 string. This string is then broken up into blocks of 650 characters. Since the image doesn't normally break up perfectly, the final block will almost certainly be less than 650 characters. These blocks are then sent individually to the cache. Once this transfer of data is completed, it is saved and sent to the client.

   It is important to note, if something happens to the server before the transfer is complete, the cache will save the blocks sent to it and keep a record of how many blocks have been sent. The next time this item is downloaded by the client, the cache will retrieve the rest of the blocks.

   Given that the blocks are sent in order, meaning that the characters at the start of the base64 string always get sent before the last. This makes keeping track of how much of a file has been sent easy as all the cache has to do is count the characters it has received and it knows how many blocks it has, and with the information sent from the server, it will also know how many is missing.

(e) Implementing option 1 would be similar to my current approach for option 2, however, for large images it would be much quicker. This is because it wouldn't need to send each block individually between the server and the cache. With option 1, images only need to be sent as one transfer which is much quicker but doesn't allow for fragmentation.

After testing my current implementation with the test files we have been given, as well as a few other test images, it is clear that the fragmentation of the files slows the server/cache transfer too much. The network bandwidth needed for this process in option two is much greater than in option one as well as the computational requirements. The computational difference between the two options is much greater as encryption of the individual chunks vs the entire image at once increases the computation load on the server and cache (because the server encrypts and the cache decrypts).