**Checklist/Other technical details**

|  |  |  |
| --- | --- | --- |
|  | **Item** | **your assignment details** |
| 1 | Names and ID numbers of Group Members | 17044923 - Jordan Drumm  15160640 - Zane Lamb  16463710 - Benjamin Upton |
| 2 | Operating System(s) used for testing your FTP server codes | Windows 10 pro |
| 3 | Compiler used | g++ 8.2.0 |
| 4 | IDE used | Sublime Text 3 |
| 5 | Successful sending of the encrypted RSA public key to the client, and decryption of the public key by the client. | Yes. |
| 6 | Correctly sending the encrypted random number (encrypted(nonce)) to the server. That also includes the server decrypting and extracting the nonce correctly. | Yes |
| 7 | Correct implementation of RSA with Cipher Block Chaining. The encryption/decryption results should be always correct for any ASCII characters. | Yes |
| 8 | Write a pseudocode of your system (i.e. specify the message format, encoding, decoding scheme, how keys are sent, generation of keys, etc.). | Yes |
| 9 | Provide a snapshot of one complete client/server interaction. For testing, use the message “**the quick brown fox jumps over the lazy dog**.” | Yes |
| 10 | Extra work done (Max Bonus of 3 marks). Implement correctly the Euclidean Algorithm and the Extended Euclidean Algorithm in combination with a big number library (make a note in your assignment submission if you did this. | Yes. The extended Euclidean algorithm was used in combination with InfInt (a big number library <https://github.com/sercantutar/infint>) to calculate a D value dependant on two randomly generated large primes. The Euclidean algorithm was implemented to calculate an E value that would be co-prime with Z. Infint was utilized so that we could have numbers larger than what an integer can hold and as such allows for the encryption to be safer. |

# Pseudo code:

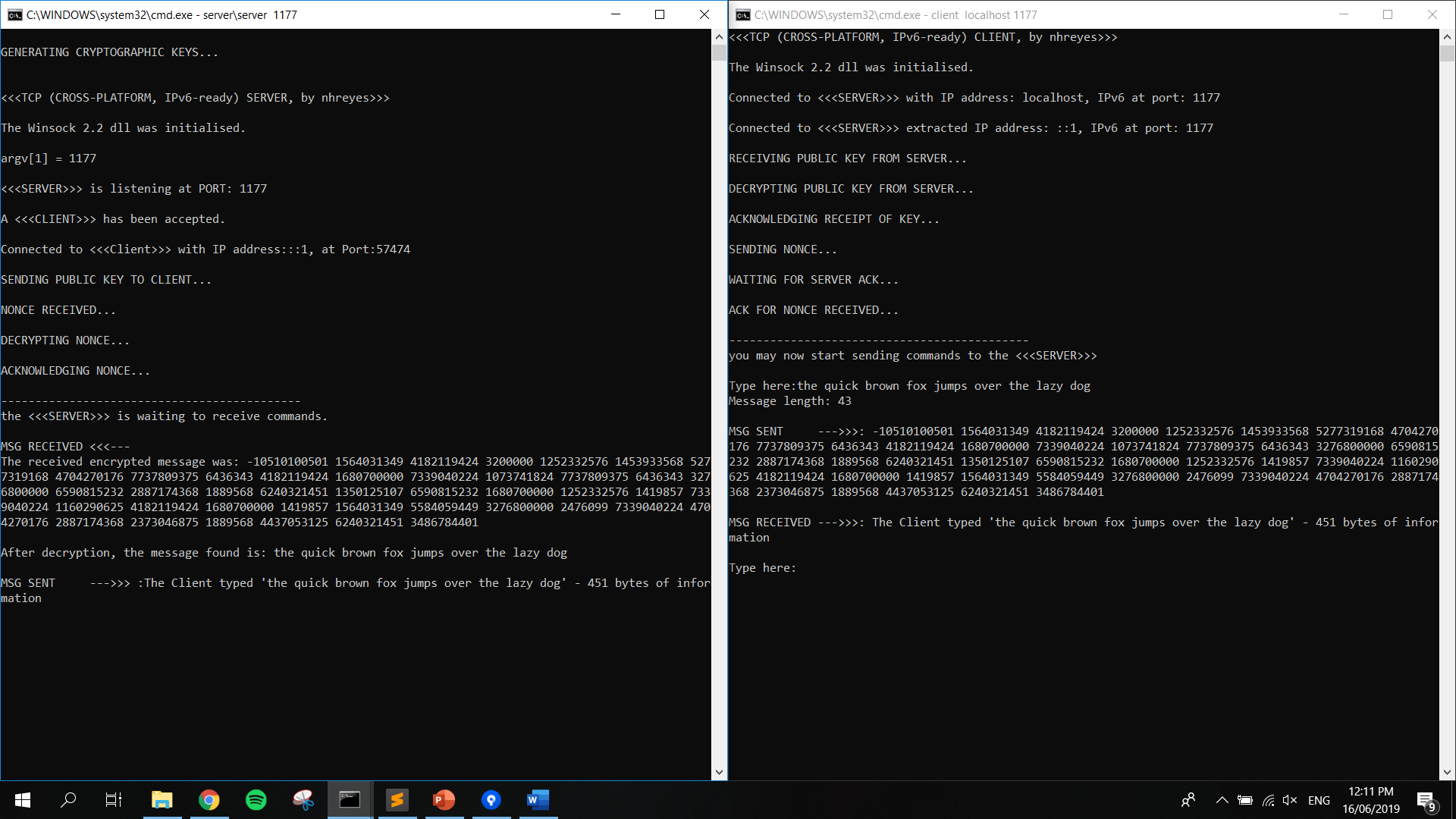
**Server**

* Generates P and Q
  + Generates two random integer numbers, multiplies them together, saves the result.
  + Repeats above step x times as desired to generate a large number
  + Checks to see if number is prime, if not, decrement it by 1 and test again
  + Repeat for Q
* Calculates N and Z
* Generates E value.
  + Starts at 2, tests to see if the GCD with Z is 1. If not, increase by 1 and re-test
  + Checks that it is not equal to P or Q and smaller than N
* Finds a suitable D value using Extended Euclidean Algorithm
* Checks that D value is correct. If not, regenerates P and Q
* Sets up socket and initialises TCP connection
* Waits for incoming connection
* Accepts client, Encrypts N and E using CA key and sends to client
* Receives nonce encrypted with N and E
* Decrypts nonce using generated D and N
* Acknowledges nonce
* Waits for message
* Receives message, decrypts it using Cypher Block Chaining
  + Extracts first token from space separated buffer
  + Saves first bit to use as the next random character
  + Decrypts character
  + XORs with nonce
  + Sets nonce to saved random character
  + Continues onto next bit
* Sends message in plaintext form to client
* Waits for new message and repeats above when received

**Client**

* Connects to Server
* Receives E and N, decrypts using CA key
* Randomly generates Nonce
* Encrypts nonce using received E and N. Sends to Server.
* Receives Nonce ACK
* Wait for input from the user
* Encrypts user using Cipher Block Chaining
  + XORs first bit with Nonce
  + Encrypts bit
  + Sets nonce to encrypted bit
  + Puts encrypted bit into space separated buffer
  + Continues onto next bit
* Sends message to Server
* Waits for plain text message from Server
* Waits for new user input and repeats above when received

# Screenshots:



# **NOTE:**

Our program will work with 300 digit prime numbers for P and Q and as such a very large N value. Whilst we can generate said prime numbers, it is very computationally expensive so in our current version we are generating relatively small random prime numbers (around 10 digits long). To test with large (300 digit long) prime numbers, simply go to lines 229 and 231 and comment in the hardcoded P and Q values. If you wish to test the large prime generation, go to line 176 and adjust the range of the for loop to be < a larger number. As this takes so long to generate, the client and server can’t be run utilizing the batch file and as such the server and client executables will need to be run separately. Decryption using very large (300 digit) keys takes a very long time (30-60+ mins) and as such testing should be done only when time allows.