|  |  |  |
| --- | --- | --- |
|  | **Item** | **Assignment details** |
| 1 | Names and ID numbers of Group Members | Dylan Cross - 15219491  Nate Fort - 15195444  Tom Sloman - 15078758 |
| 2 | Operating System | Windows 10 |
| 3 | Compiler used | GCC 4.8.1 |
| 4 | IDE used | VSCode (Text Editor) |
| 5 | Required Functionalities  Send Public key encrypted with dCA  Send Nonce encrypted with public key  Encrypt text with RSA w/CBC | Yes  Yes  Yes |
| 6 | Snap shots of sample interactions: | Yes |
| 7 | Extra work done (Bonus): | No |

Note: The RSA w/CBC would fail to successfully decrypt a character if the previous encrypted character was close (within 10-20) of the session’s N value. To solve this we divide the previous C value by 2 when encrypting/decrypting.   
Example: if (c[i-1] > n – 20) then when encrypting/decrypting c[i] use (c[i-1]/2) as the cbc value

**Format of interaction between server an client:**

On connection, client asks server for the E and N (public key) for the session.  
 🡪 SEND RSA\r\n  
The server responds with the public key encrypted with the dCA private key.  
 🡨 RSA E:*123*;N:*123*;\r\n  
The Client decrypts with the public key eCA and sends an acknowledgement.  
 🡪 RSA ACK\r\n  
The Client generates Nonce (which is less than N) which is then encrypted with the session’s public key.  
 🡪 RSA NONCE:*123*\r\n  
The Server sends back an Acknowledgement that nonce is ok.  
 🡨ACK 220 NONCE OK\r\n  
The Client sends the text encrypted with the session’s public key.  
 🡪C:1390;2682;1315;1823;2853;  
The Server sends back the decrypted text  
 🡨hello

Screenshots of interactions: