|  |  |  |
| --- | --- | --- |
|  | **Item** | **Assignment Details** |
| **1** | Name & IDs | **James Waddell (16379344)**  **Taylor Bennett (16105740)** |
| **2** | Operating System | Windows 10 |
| **3** | Compiler Used | GCC v8.2 |
| **4** | IDE Used | SublimeText 3 |
| **5** | Documentation | Yes (this is it) |
| **7** | Screenshots | Yes (see documentation folder) |
| **8** | Program Functionality | |  |  |  | | --- | --- | --- | | **Functionality** | **Implementation**  **Level** | **Marks** | | Sending of the encrypted RSA public key to the client, and decryption of the public key by the client | **FULL** | **2** | | Sending the encrypted random number (**encrypted**(**nonce**)) to the server. That also includes the server decrypting and extracting the **nonce** correctly. | **FULL** | **2** | | Implementation of RSA with Cipher Block Chaining (The encryption/decryption results should be always correct for any ASCII characters. Make sure that the combination of the **selected RSA keys** and **nonce** will not have any problem encrypting and decrypting messages.) | **FULL** | **2** | |
| **9** | Bonus Marks | |  |  |  | | --- | --- | --- | | **Functionality** | **Implementation**  **Level** | **Marks** | | **Euclidean + ExtendedEuclidean Algorithm** | **FULL** | **-** | | **Big Number Library** | **NONE** | **-** | |

**Pseudocode**

|  |  |  |
| --- | --- | --- |
| **Server** |  | **Client** |
| Start server | **-->** | Client connects |
| Server has CA private key, which it uses to encrypt the server’s public key (e, n) that is to be sent to the client.  Server’s key generation for P, Q, Z, N, E values are all generated with their own functions.  Then, dPrivateKey is calculated by calculateD(e, z), which uses the extended Euclidean algorithm to gather the result.  Encryption on the server is achieved by the encryptionDecryption() function, which uses RSA encryption. | **-->** | Client receives the encrypted server public key, which it decrypts using the CA public key. |
| Server receives client’s ACK 226 | **<--** | Client Sends ACK 226 public key received |
| Server receives encryptedNonce, to which it decrypts and stores. | **<--** | Client generates nonce using rand()  Client encrypts this randomly generated nonce using encryptionDecryption() algorithm (RSA)  Client then sends encryptedNonce to the server (straight after the ACK) |
| Server sends ACK for the Nonce | **-->** | Client receives client’s ACK 226 |
|  |  | Client hangs waiting for user input |
| Server receives the block of encrypted characters and proceeds to decrypt them.  Data is decrypted one character at a time using reverseRSA\_CBC() function, the result of this is a long.  Then the result is printed to the console and stored in the decryptedBuffer array. | **<--** | Once the user presses the return key, the client takes any user input and encrypts it using RSA & CBC.  Data is encrypted one character at a time using the calculateRSA\_CBC() function.  Then is concatenated to the send\_buffer with spaces between each encrypted character.  The send\_buffer is sent to the server. |
| Server sends the decryptedBuffer to the client in plain text.  (this is the server echoing what the client originally types) | **-->** | Client received the echoed message and prints it to console. |
| Server hangs waiting for message/command from client. If client disconnects, the server shuts down. |  | Client hangs waiting for input, to repeat the RSA\_CBC process. |