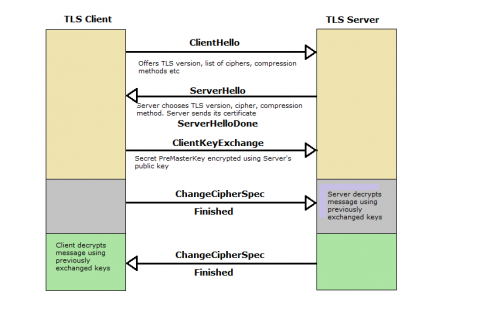
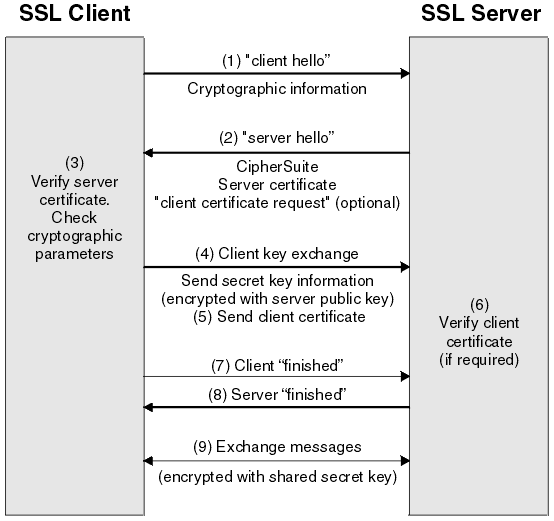
Client and Server Handshake





The TLS version is negotiated initially by the client (Client Hello message) specifying the **highest** version that it supports among other parameters (cipher parameters, etc.). Text from RFC 5246, TLS v1.2:

**client\_version:** The version of the TLS protocol by which the client wishes to communicate during this session. **This SHOULD be the latest (highest valued) version supported by the client**. For this version of the specification, the version will be 3.3 (see Appendix E for details about backward compatibility).

After the server receives the Client Hello it sends the Server Hello with the chosen SSL/TLS version among other chosen parameters based on the Client Hello information. Text from RFC 5246, TLS v1.2:

**server\_version:** This field will contain the lower of that suggested by the client in the client hello and the highest supported by the server. For this version of the specification, the version is 3.3. (See Appendix E for details about backward compatibility.)

I would say the problem is that your Client application does not accept TLS version downgrade fallback during the negotiation and that's why it works when disabling TLSv1.1 and TLSv1.2. Check the bold text in the again RFC 5246, Appendix E:

A TLS 1.2 client who wishes to negotiate with such older servers will send a normal TLS 1.2 ClientHello, containing { 3, 3 } (TLS 1.2) in  
ClientHello.client\_version. If the server does not support this  
version, it will respond with a ServerHello containing an older  
version number. If the client agrees to use this version, the  
negotiation will proceed as appropriate for the negotiated protocol.

**If the version chosen by the server is not supported by the client (or not acceptable), the client MUST send a "protocol\_version" alert  
message and close the connection.**

An easy approach to debug this, could be to check the logs of both server and application (you did not specify which applications you're using) in verbose mode so you can see the negotiation messages. Other more advanced may be to put wirehark/tcpdump to check the negotiation parameters. This should be feasible without too much hassle since these information does not travel encrypted.