By Talal Shafei 28/05/2024

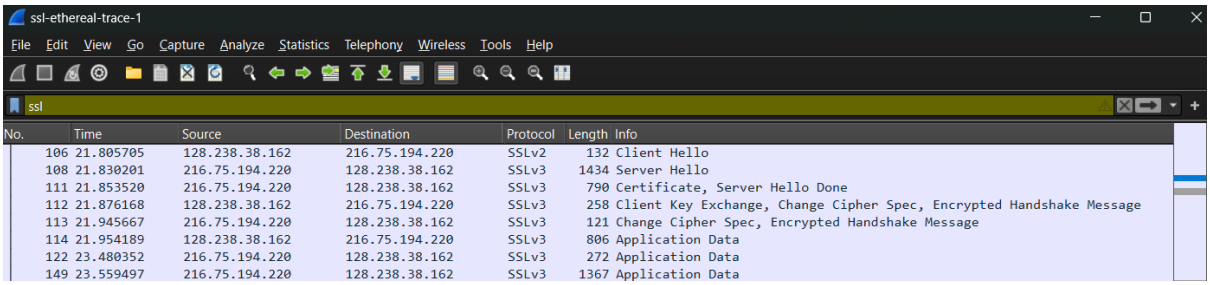
Wireshark Lab

Note

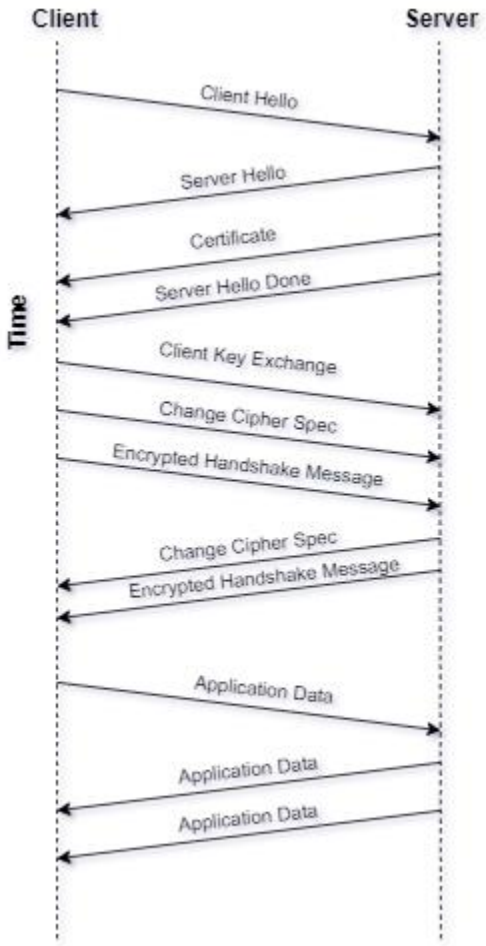
using the given trace ssl-etheral-trace-1

Solution

1. For each of the first 8 Ethernet frames, specify the source of the frame (client or server), determine the number of SSL records that are included in the frame, and list the SSL record types that are included in the frame. Draw a timing diagram between client and server, with one arrow for each SSL record.



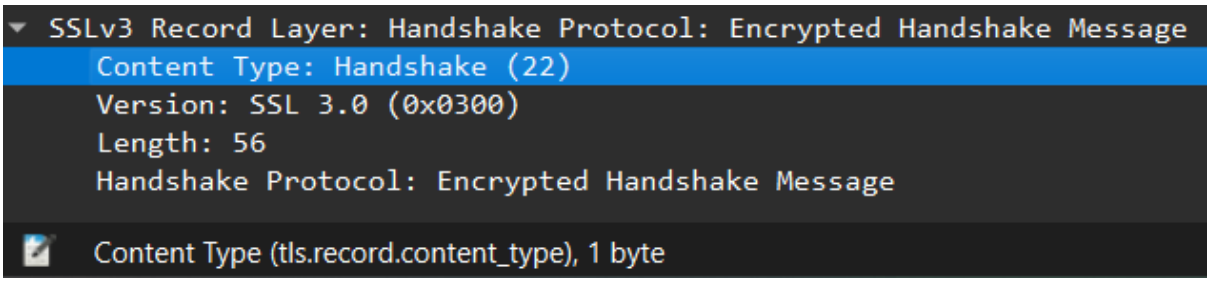
#	Frame no.	Source IP	Source name	No. SSL records	SSL type
1	106	128.238.38.162	Client	1	Client Hello
2	108	216.75.194.220	Server	1	Server Hello
3	111	216.75.194.220	Server	2	Certificate, Server Hello Done
4	112	128.238.38.162	Client	3	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
5	113	216.75.194.220	Server	2	Change Cipher Spec, Encrypted Handshake Message
6	114	128.238.38.162	Client	1	Application Data
7	122	216.75.194.220	Server	1	Application Data
8	149	216.75.194.220	Server	1	Application Data

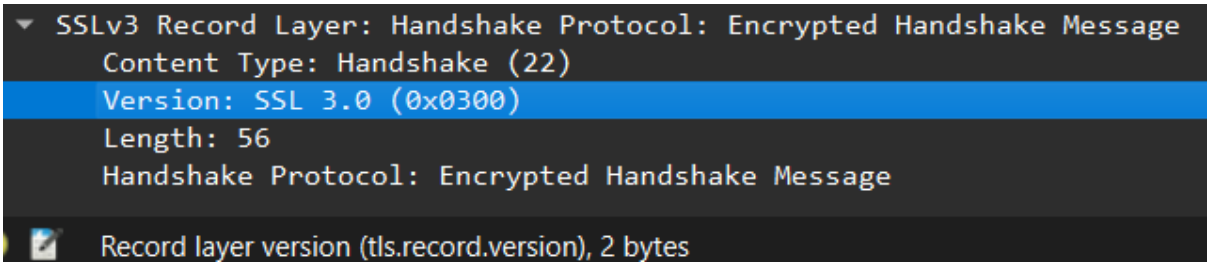


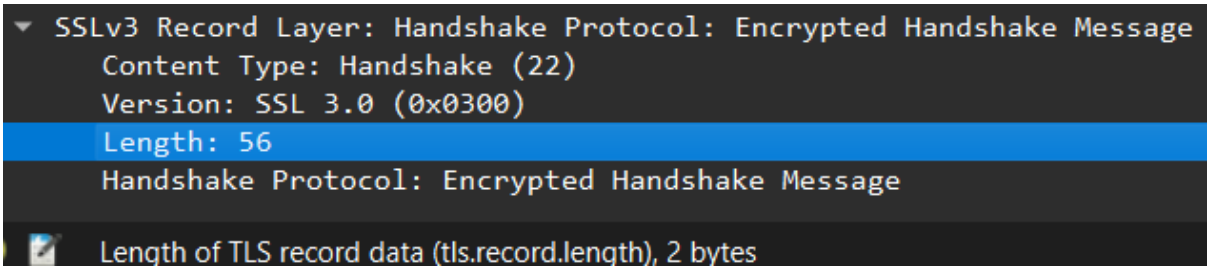
2. Each of the SSL records begins with the same three fields (with possibly different values). One of these fields is “content type” and has length of one byte. List all three fields and their lengths.

   We can check their length by pressing on the field and their length will appear at the bottom of the window, as shown in the images below.

Field name	Size (bytes)
Content Type	1
Version	2
Length	2





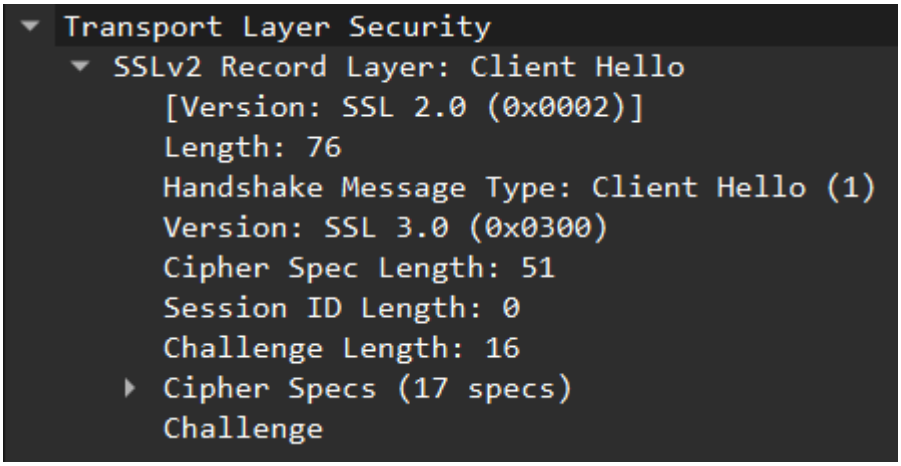


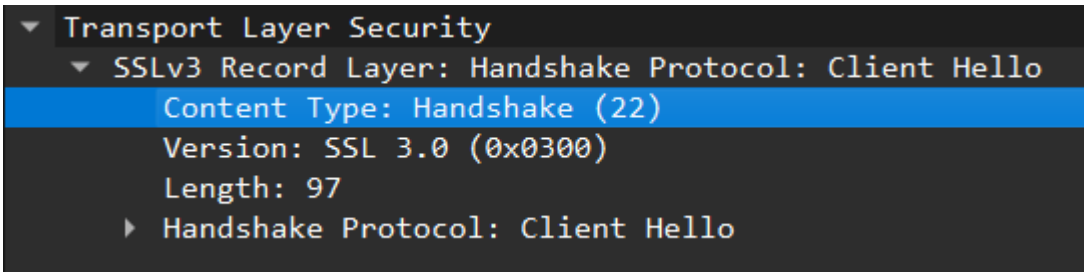
Client Hello Record: packet no. 106

3. Expand the ClientHello record. (If your trace contains multiple ClientHello records, expand the frame that contains the first one.) What is the value of the content type?

   Note: The first ClientHello is in SSLv2

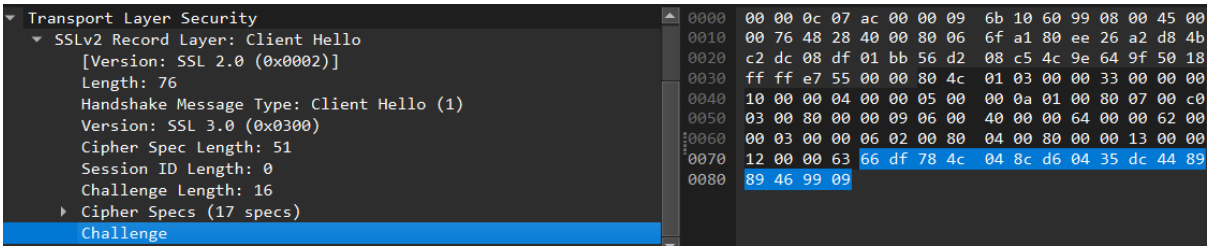
   There is no content type in SSLv2, but from the 'Handshake Message Type' it is obvious its type is Handshake, in the following image you can see the second ClientHello in SSLv3 where the content type is explicitly mentioned as 'Handshake (22)' and its value is 22.





4. Does the ClientHello record contain a nonce (also known as a “challenge”)? If so, what is the value of the challenge in hexadecimal notation?

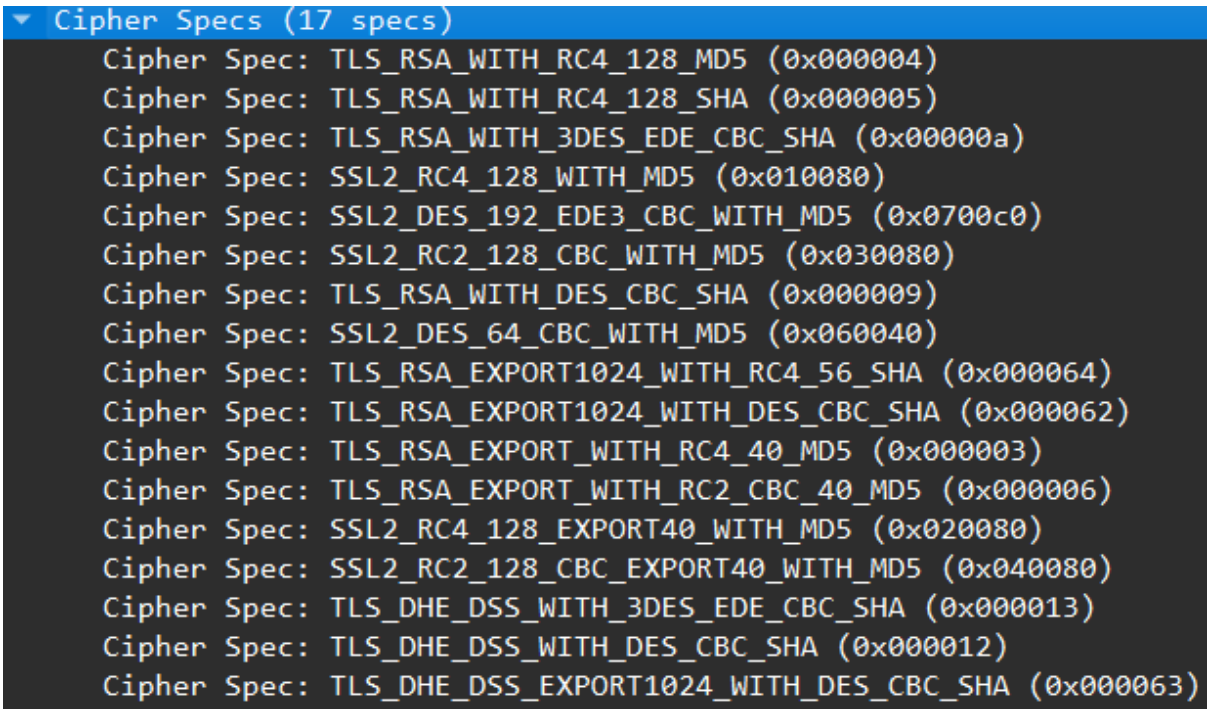
   Yes, and it does contain and it’s 16 bytes in length
   Value in hex: 66df784c048cd60435dc448989469909



5. Does the ClientHello record advertise the cyber suites it supports? If so, in the first listed suite, what are the public-key algorithm, the symmetric-key algorithm, and the hash algorithm?
   Yes, below you can see the client sharing with the server what are the algorithms it can support.

   First suite: TLS_RSA_WITH_RC4_128_MD5

Algorithm	Value
Public-key	RSA
Symmetric-key	RC4 128 (Rivest Cipher 4)
Hash	MD5

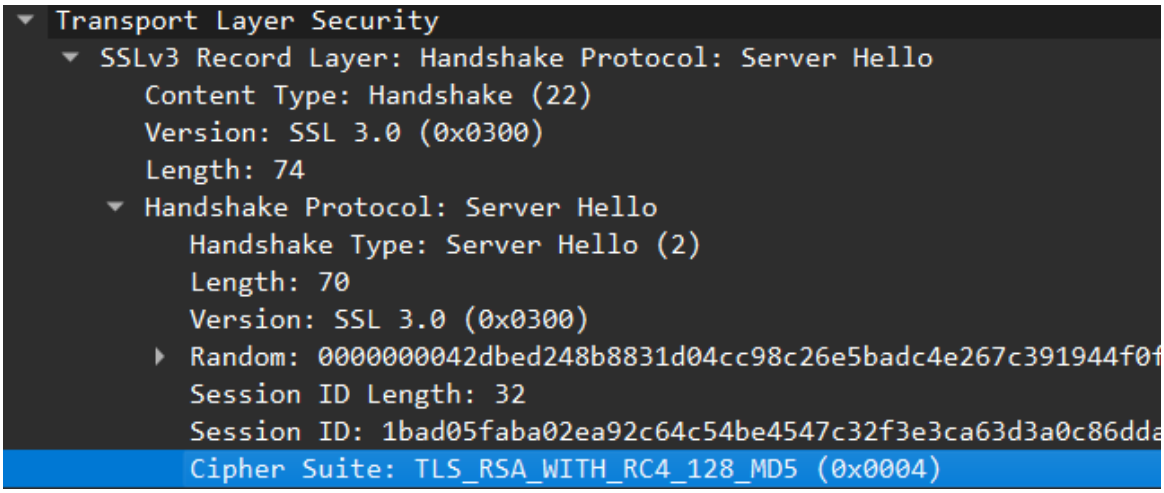


ServerHello Record: packet no. 108

6. Locate the ServerHello SSL record. Does this record specify a chosen cipher suite? What are the algorithms in the chosen cipher suite?

   Yes, it does contain.
   Chosen Cipher Suite: TLS_RSA_WITH_RC4_128_MD5

Algorithm	Value
Public-key	RSA
Symmetric-key	RC4 128 (Rivest Cipher 4)
Hash	MD5

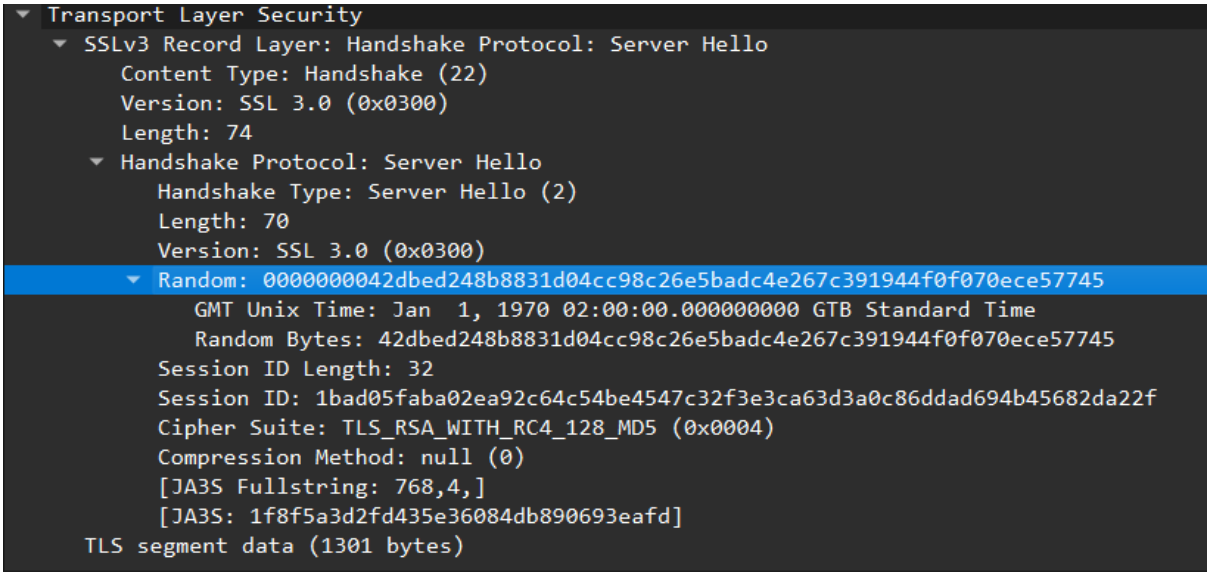


7. Does this record include a nonce? If so, how long is it? What is the purpose of the client and server nonces in SSL?

   Yes, and it is called 'Random'
   Size: 32 bytes
   Value in hex:
   0000000042dbed248b8831d04cc98c26e5badc4e267c391944f0f070ece57745
   Purpose:

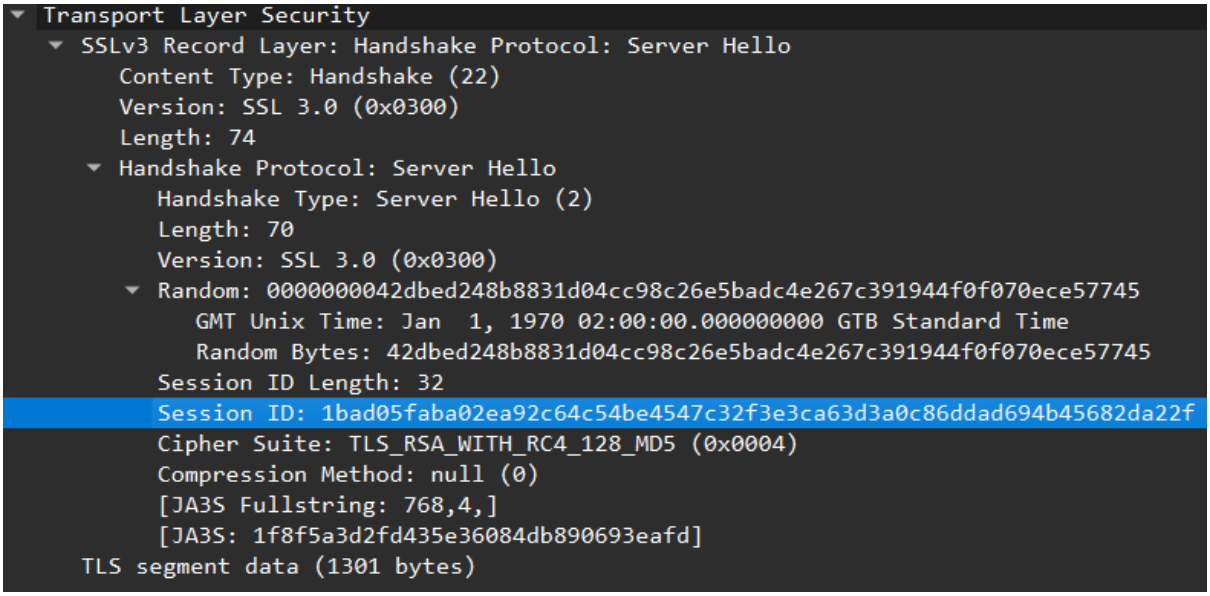
   1. Prevent replay attacks: nonce confirm that the session is unique so if someone try to replay an old message, their nonce will differ, and server will reject it.

   2. For deriving keys: key generation for both client and server, so even if the same pre-master key is used, each session will have a different session-key due to nonce randomness.



8. Does this record include a session ID? What is the purpose of the session ID?

   Yes Session ID is present in the ServerHello, and it is used to restore or resume a previous connection, and this can be done when the client sends the Session ID with ClientHello packet when it's reconnecting and if the server remembers it will reuse the old parameters instead of establishing a new connection and going through the entire handshake process again, so Session ID provides efficiency when it comes to avoid extra computational load.

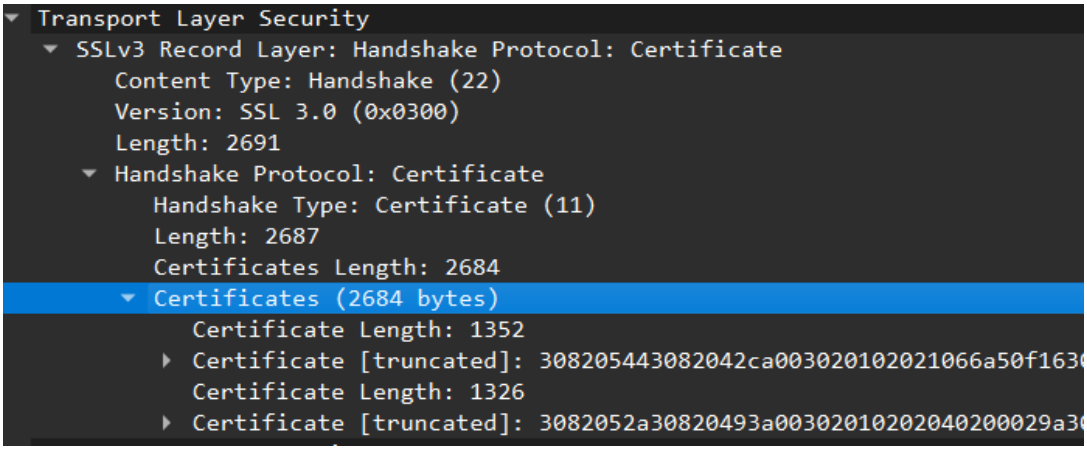


9. Does this record contain a certificate, or is the certificate included in a separate record. Does the certificate fit into a single Ethernet frame?

   The Certificate is included in a separate record (no. 111).

   There are 2 certificates, and their total size is 2684 bytes, which does not fit in one Ethernet frame that has a maximum size of 1518 bytes and 1500 bytes of them for payload, but each Certificate separately can fit.

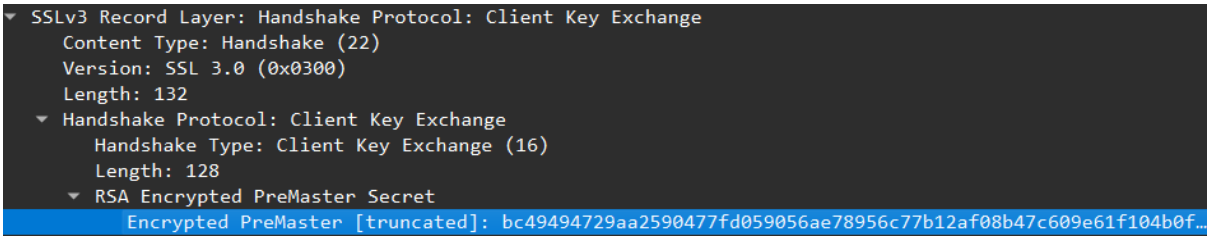
   Ethernet frame size was based according to this web page.



Client Key Exchange Record: packet no. 112

10. Locate the client key exchange record. Does this record contain a pre-master secret? What is this secret used for? Is the secret encrypted? If so, how? How long is the encrypted secret?

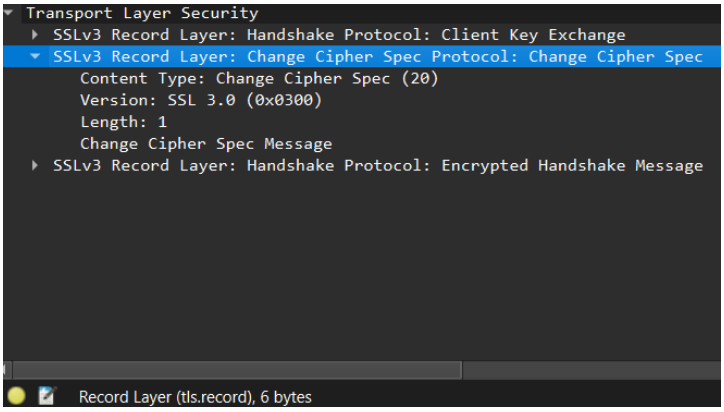
    - Yes, it does contain pre-master secret.
    - It is used for generating the master secret key by the server that will be used later to generate the session keys.
    - Yes, the client encrypted it according to RSA (chosen earlier in the handshake) with the server's public key to ensure only the server can decrypt the secret.
    - Length of the pre-master secret is 128 bytes.



Change Cipher Spec Record (sent by client) and Encrypted Handshake Record:

11. What is the purpose of the Change Cipher Spec record? How many bytes is the record in your trace?

    Purpose: The Change Cipher Spec is the point in the handshake where the client and the server confirm the chosen Cipher Suite and switch to communicate with encrypted messages for that session.
    The record size is 6 bytes.



12. In the encrypted handshake record, what is being encrypted? How?

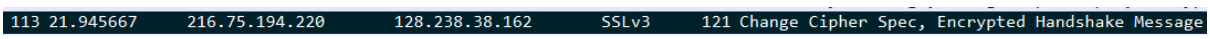
    It contains a hash that represents message authentication code (MAC) of all the previous messages, and this to ensure integrity of the handshake process between server and client and it confirms that both parties are aware of the security parameters that they will be using during the communication.

    It's encrypted using the session key by the Symmetric-key algorithm that was chosen earlier and, in our case, it is RC4 128.

13. Does the server also send a change cipher record and an encrypted handshake record to the client? How are those records different from those sent by the client?

    Yes, server also does send, so the client can confirm that it is communicating the intended server, and the secure encrypted connection was established correctly and after it the client starts send and expect receive encrypted messages of type Application Data.

    Here the server sends both Change Cipher Spec and Encrypted Handshake Message in one packet.



Application Data

14. How is the application data being encrypted? Do the records containing application data include a MAC? Does Wireshark distinguish between the encrypted application data and the MAC?

    It is being encrypted using the Symmetric-key algorithm that was agreed upon during the handshake and here it's RC4.

    Yes, they include Message Authentication Code (MAC) which is a digest for the data before encryption using the hash chosen from before, which is MD5 in this example.

    MAC is appended at the beginning of the payload and its used to compare it with the digest of the decrypted data make sure the data was not tampered and ensure integrity.

    No, Wireshark can't distinguish between MAC and the encrypted message.

15. Comment on and explain anything else that you found interesting in the trace.

    One thing that amazed me is how I never thought of how much is going through behind the scenes of SSL/TLS, opening a browser and going through my favourite websites looked like very smooth and simple operation, but now I realized how much back and forth my laptop and the server communicate to establish a secure connection while relying on very complicated algorithms.

    It is very interesting that using an insecure channel of communication we still were able to convert it completely to a secure encrypted channel.

    Also, one thing I realized is when the server returned the Certificates in packet no.111 it was not very realistic because even though they contain public keys still what confirm that they were sent by the server and not a middleman?? That's why I guess a more realistic approach should have been using trusted third party such as Certificate Authority (CA).