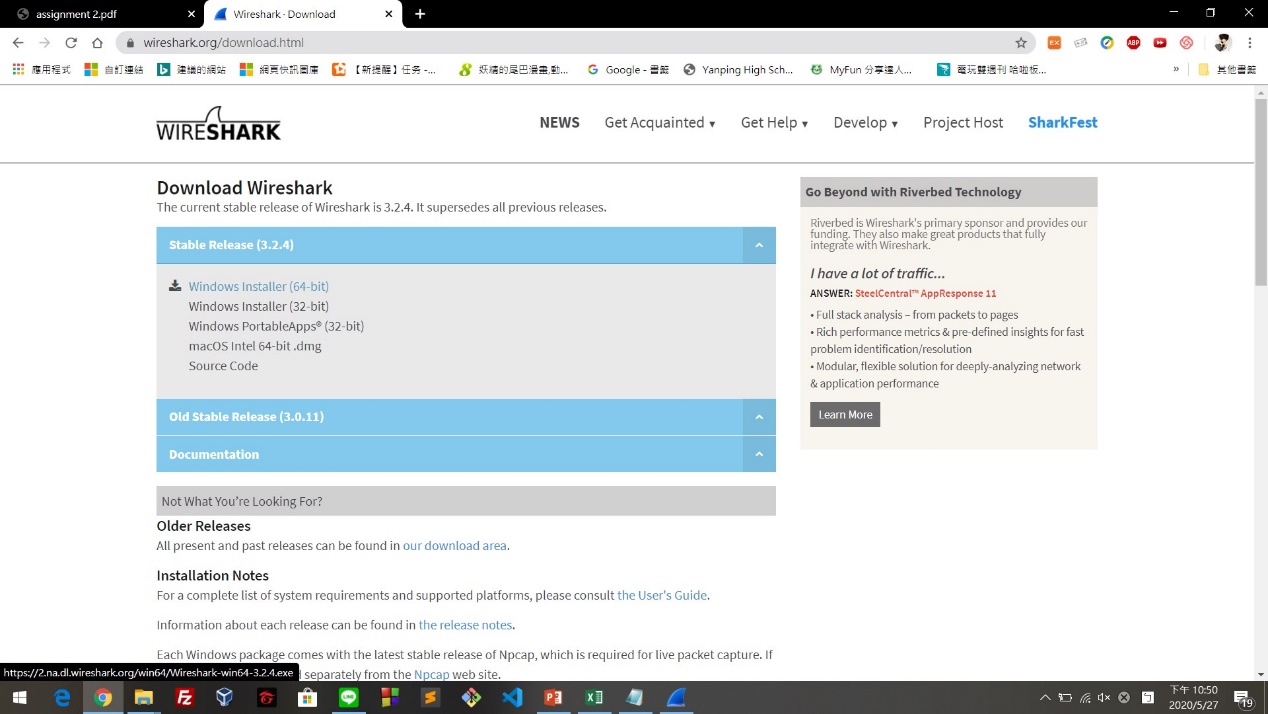
Assignment 2’s Report

**0616098 黃秉茂**

**Task 1. 安裝sniffer(Wireshark)**

到<https://www.wireshark.org/download.html>

點擊Windows Installer (64-bit) (因為現在電腦通常是64-bit)



接著就會出現Wireshark的安裝檔



雙擊後，一直點擊 "next"、"下一步" 等字眼就安裝完成了



**Task 2. 用wget來瀏覽「支援HTTPS的網頁」**

因為Windows本身無內建wget，需要自己上網下載

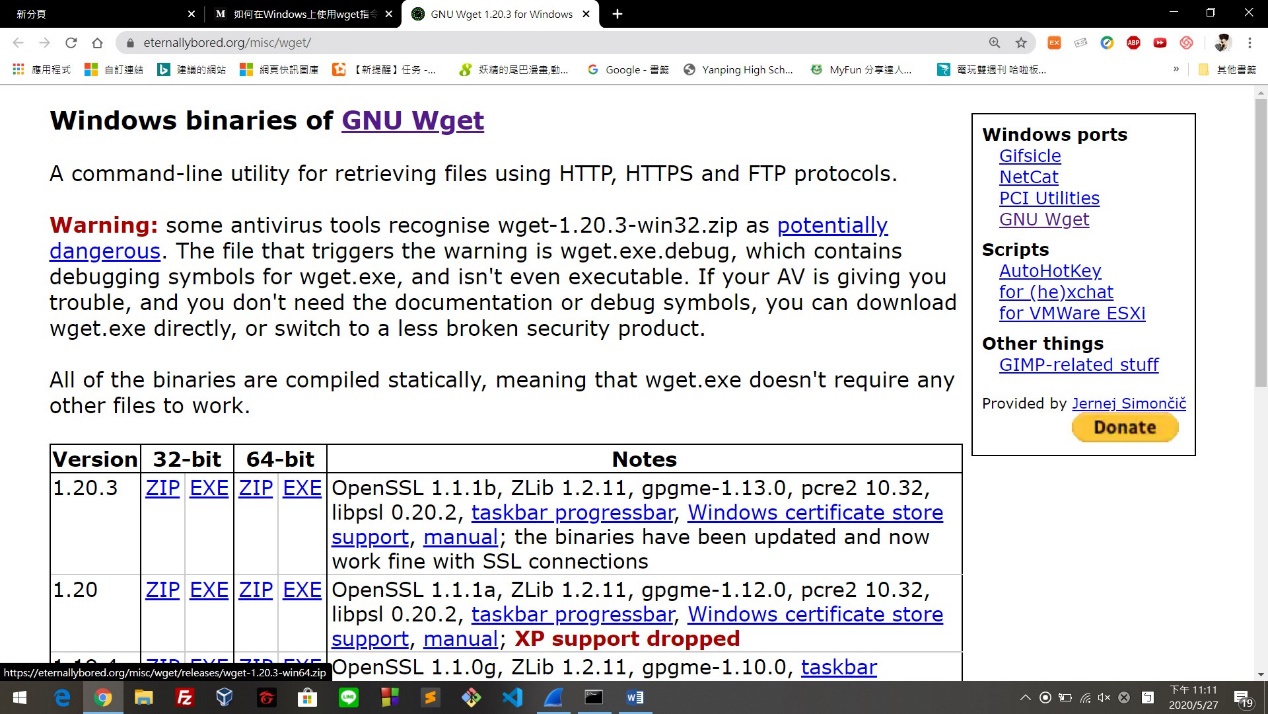
安裝wget：

(參考：如何在Windows上使用wget指令下載檔案？

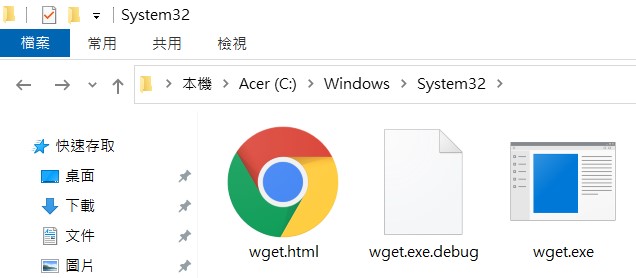
<https://medium.com/@yanweiliu/%E5%A6%82%E4%BD%95%E5%9C%A8windows%E4%B8%8A%E4%BD%BF%E7%94%A8wget%E6%8C%87%E4%BB%A4%E4%B8%8B%E8%BC%89%E6%AA%94%E6%A1%88-93730d4b4915>)

到<https://eternallybored.org/misc/wget/>

下載wget檔案(Version 1.20.3 的64-bit的ZIP)



將下載的ZIP檔解壓縮到C:\Windows\System32



基本上有出現wget.exe就代表成功了

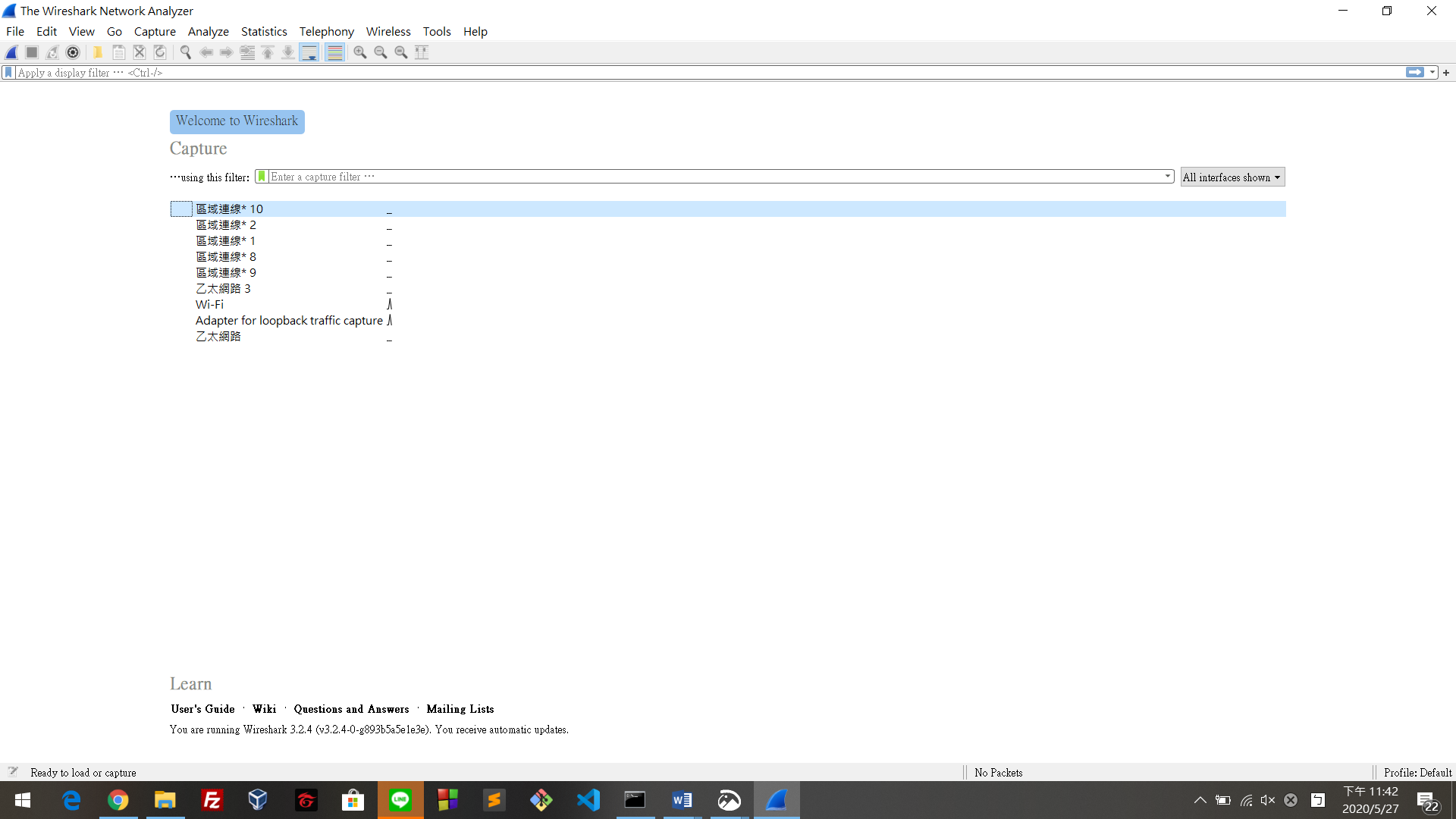
測試wget安裝：

開啟CMD，輸入wget --help (有出現這些代表wget安裝成功)



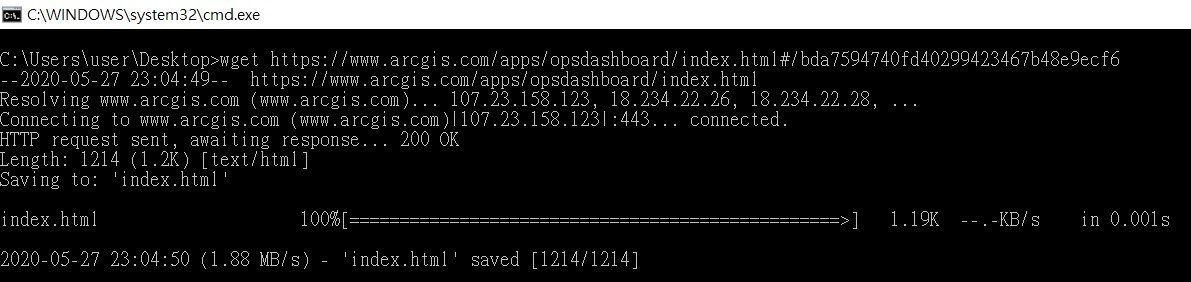
用wget瀏覽網頁並用Wireshark側錄：

開啟Wireshark點擊Wi-Fi開始側錄(因為我是用Wi-Fi連線)：

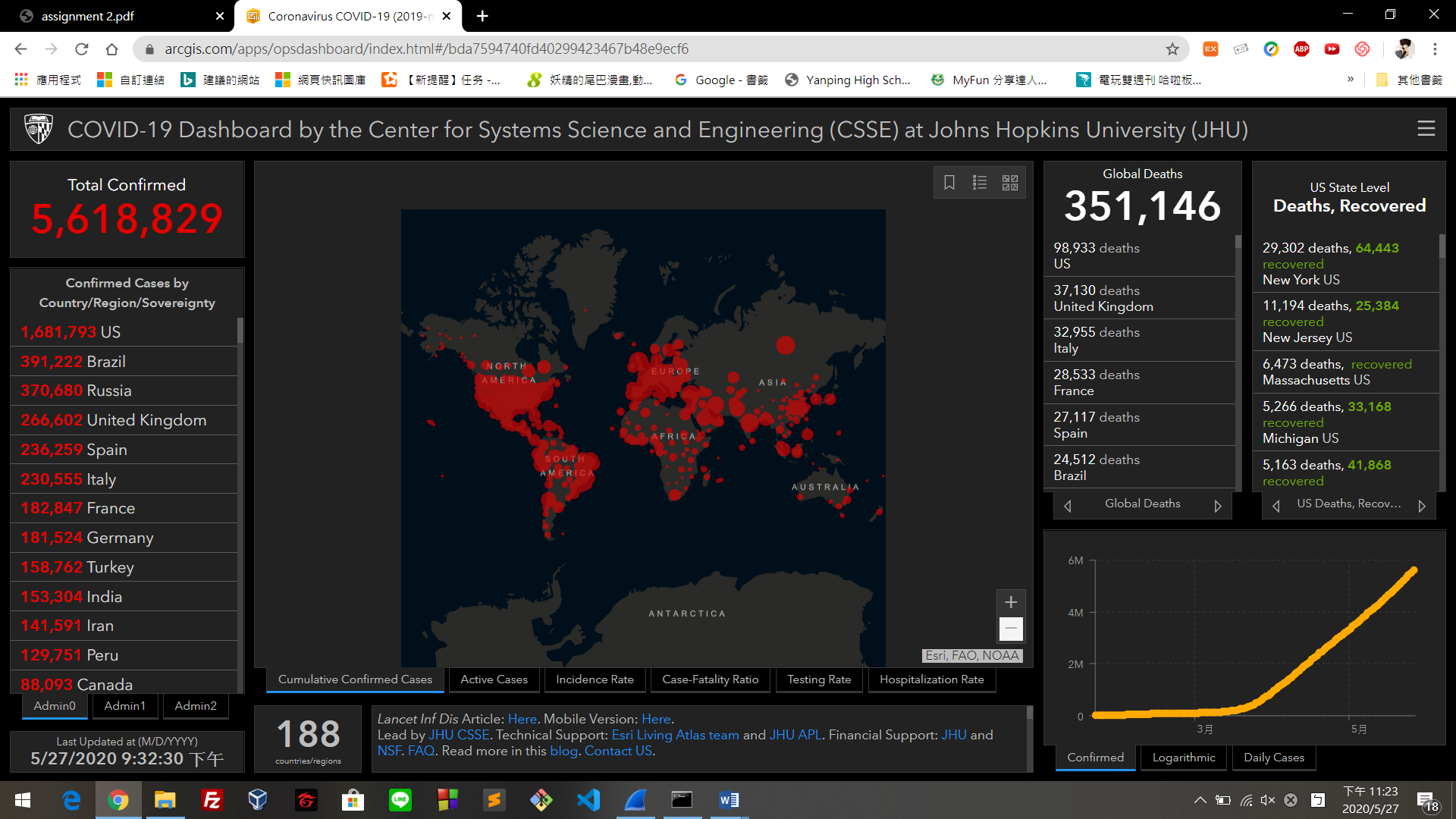


在cmd輸入wget + 目標網址

(<https://www.arcgis.com/apps/opsdashboard/index.html#/bda7594740fd40299423467b48e9ecf6>)

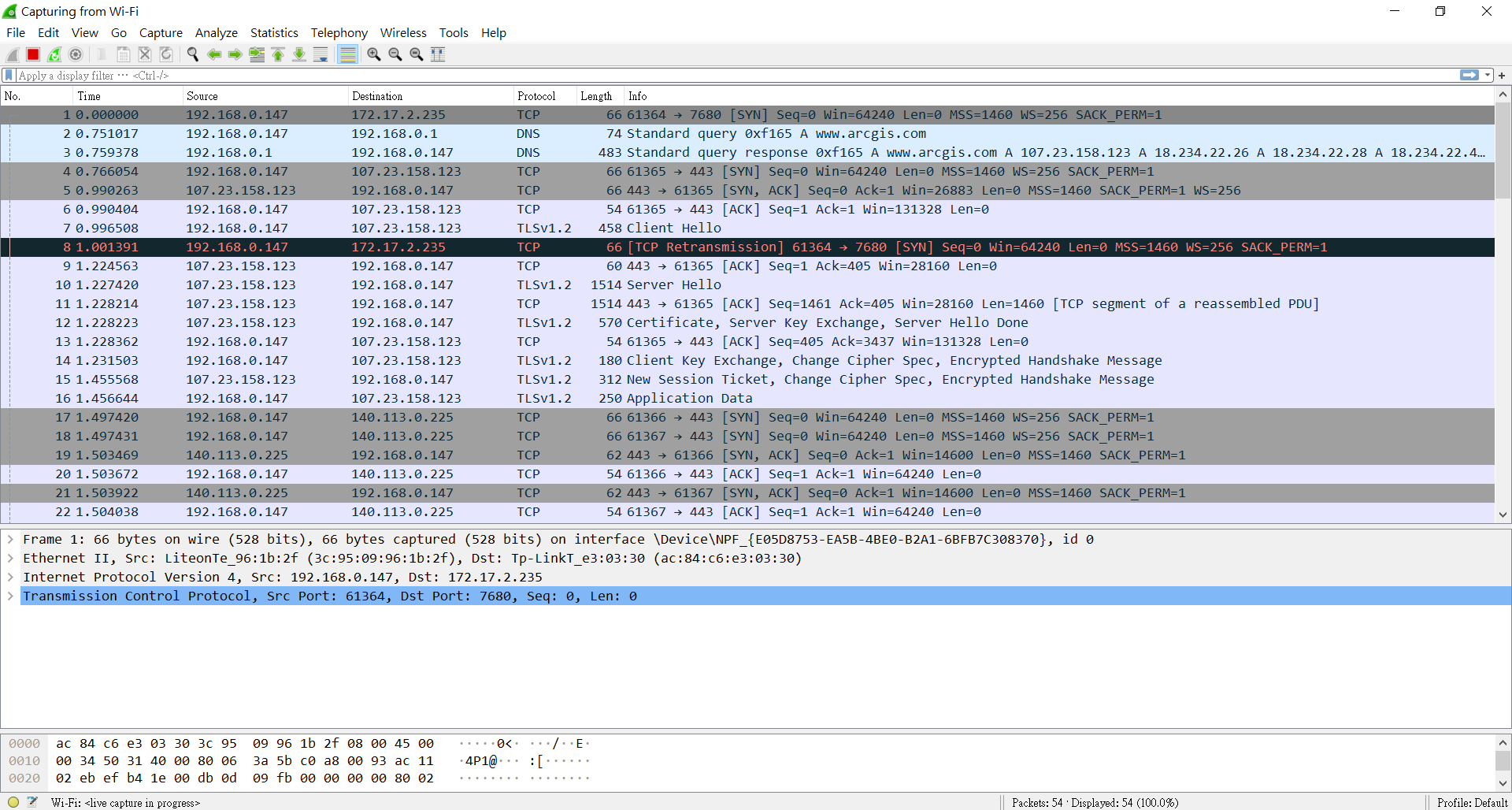


<https://www.arcgis.com/apps/opsdashboard/index.html#/bda7594740fd40299423467b48e9ecf6>的樣貌

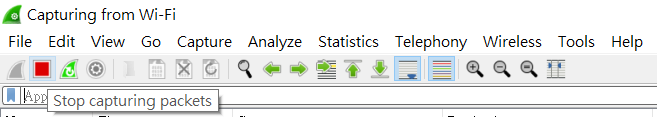


之後切到Wireshark，便會出現以下畫面

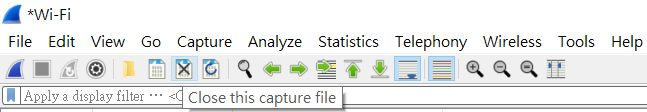
(抓取到的側錄的封包的資訊)



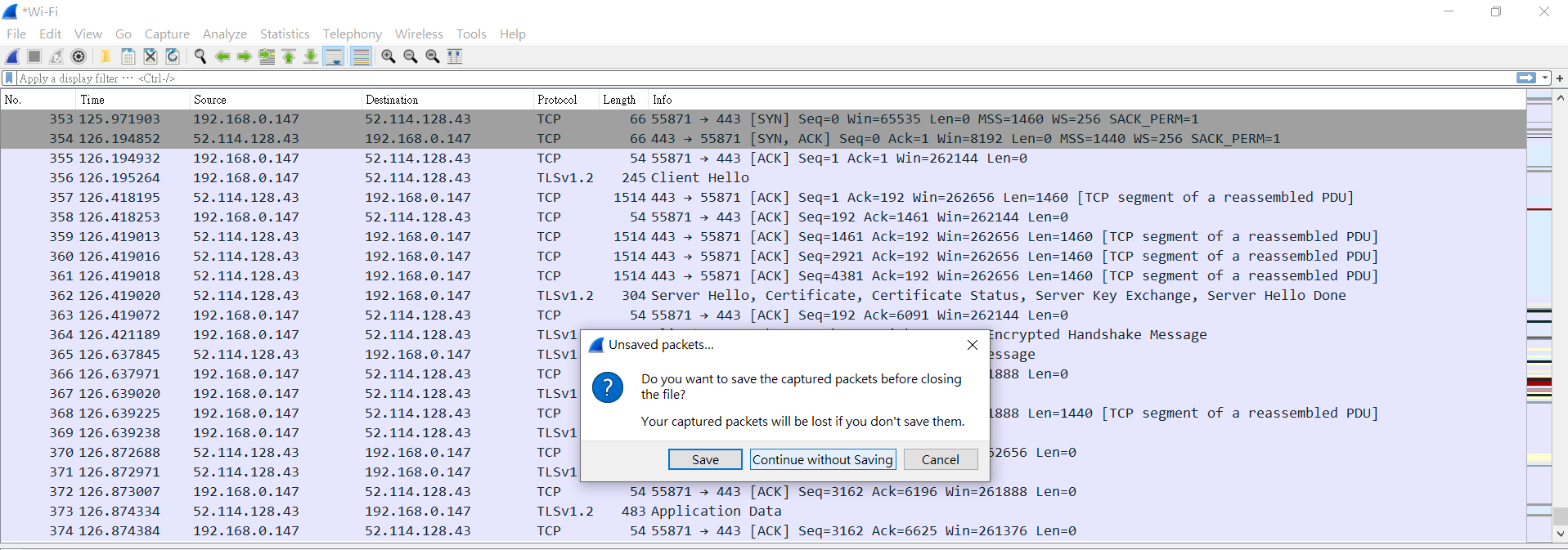
點擊紅色方框，就能停止抓取封包及側錄



如果要清除抓取的封包，點擊黑色叉叉，將所有抓到的封包清掉



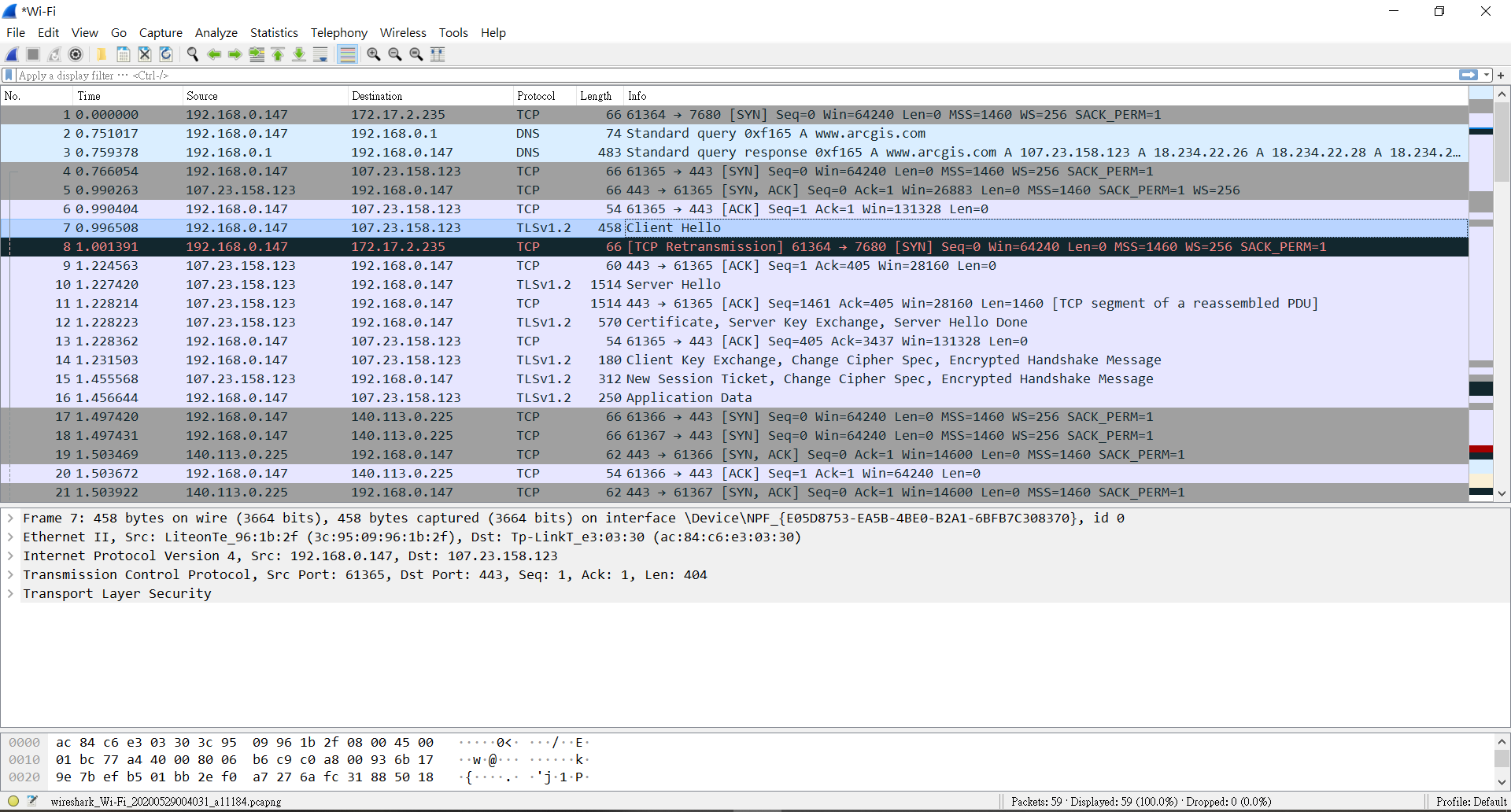
之後點擊"Continue without Saving"，就能清除所有抓取道的封包的資訊



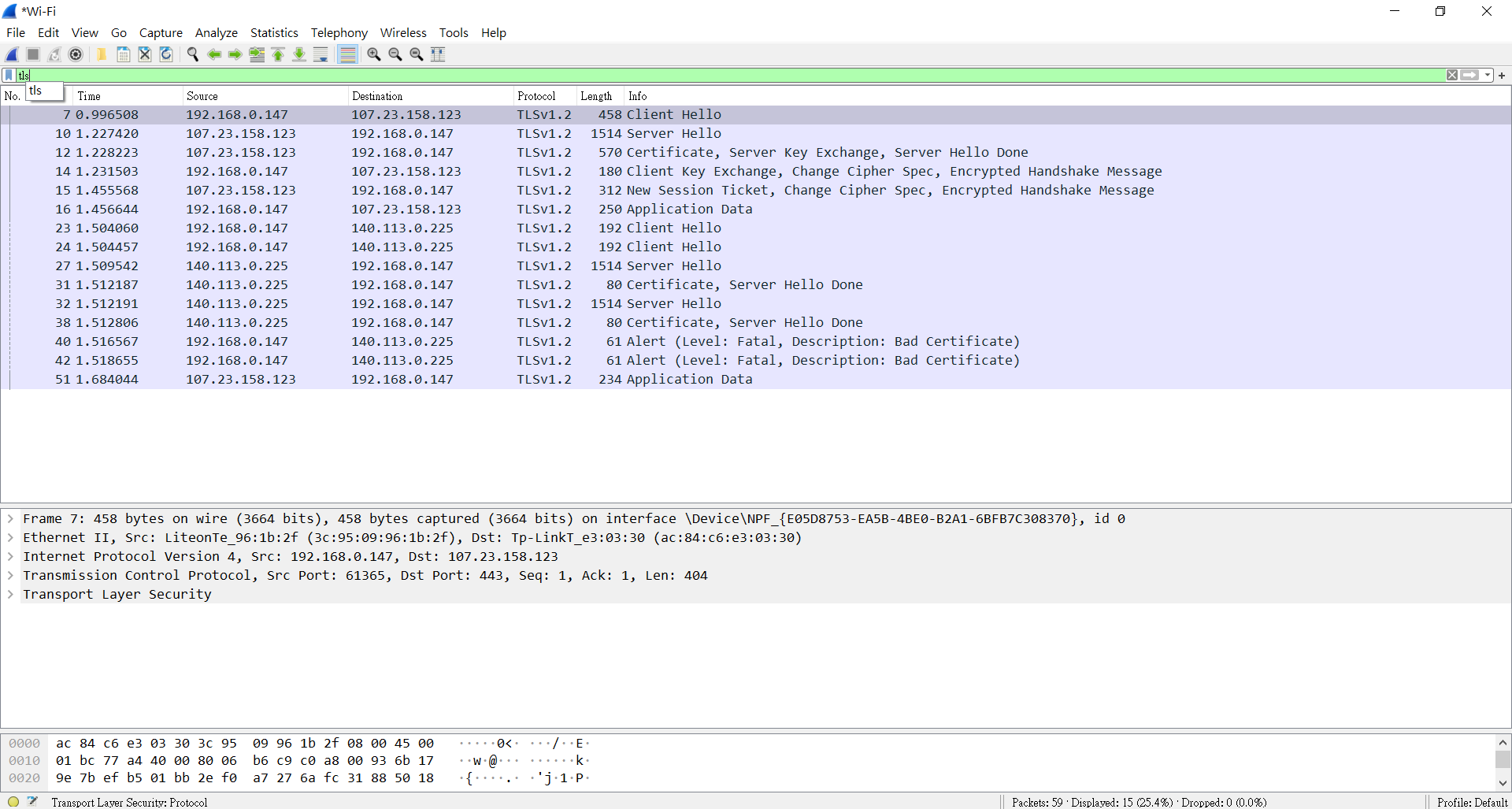
**Task 3. TLS handshake**

其中第7、10、12、14、15依序為TLS handshake 的步驟

(Client Hello, Server Hello, Certificate, Server Key Exchange, Server Hello Done, Client Key Exchange, Change Cipher Spec, Encrypted Handshake Massage(Finished), Change Cipher Spec, Encrypted Handshake Massage(Finished))



因為TLS handshake的Protocol是TLS，所以在filter上輸入”tls”，篩選出較相關的較乾淨的 packet trace

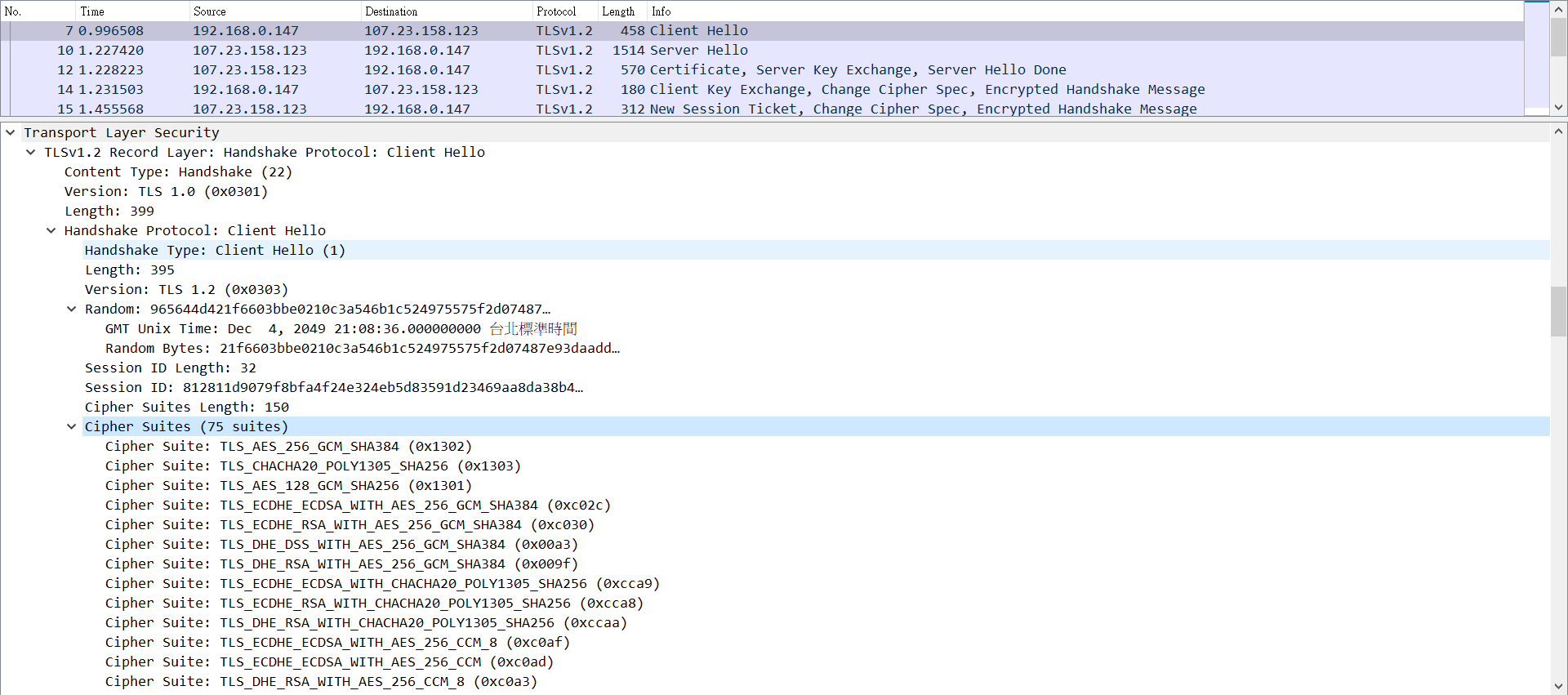


1. Initial Client to Server Communication：

No. 7 Client Hello：

Typically, the first message in the TLS Handshake is the client hello message which is sent by the client to initiate a session with the server.

When a client first connects to a server, it is required to send the ClientHello as its first message. The client can also send a ClientHello in response to a HelloRequest or on its own initiative in order to renegotiate the security parameters in an existing connection.



Record Layer Version: The TLS protocol version number that communication uses.

inner layer Version: The TLS protocol version number that the client wants to use for communication with the server. This is the highest version supported by the client.

Random: A 32-byte pseudorandom number that is used to calculate the Master secret (used in the creation of the encryption key).

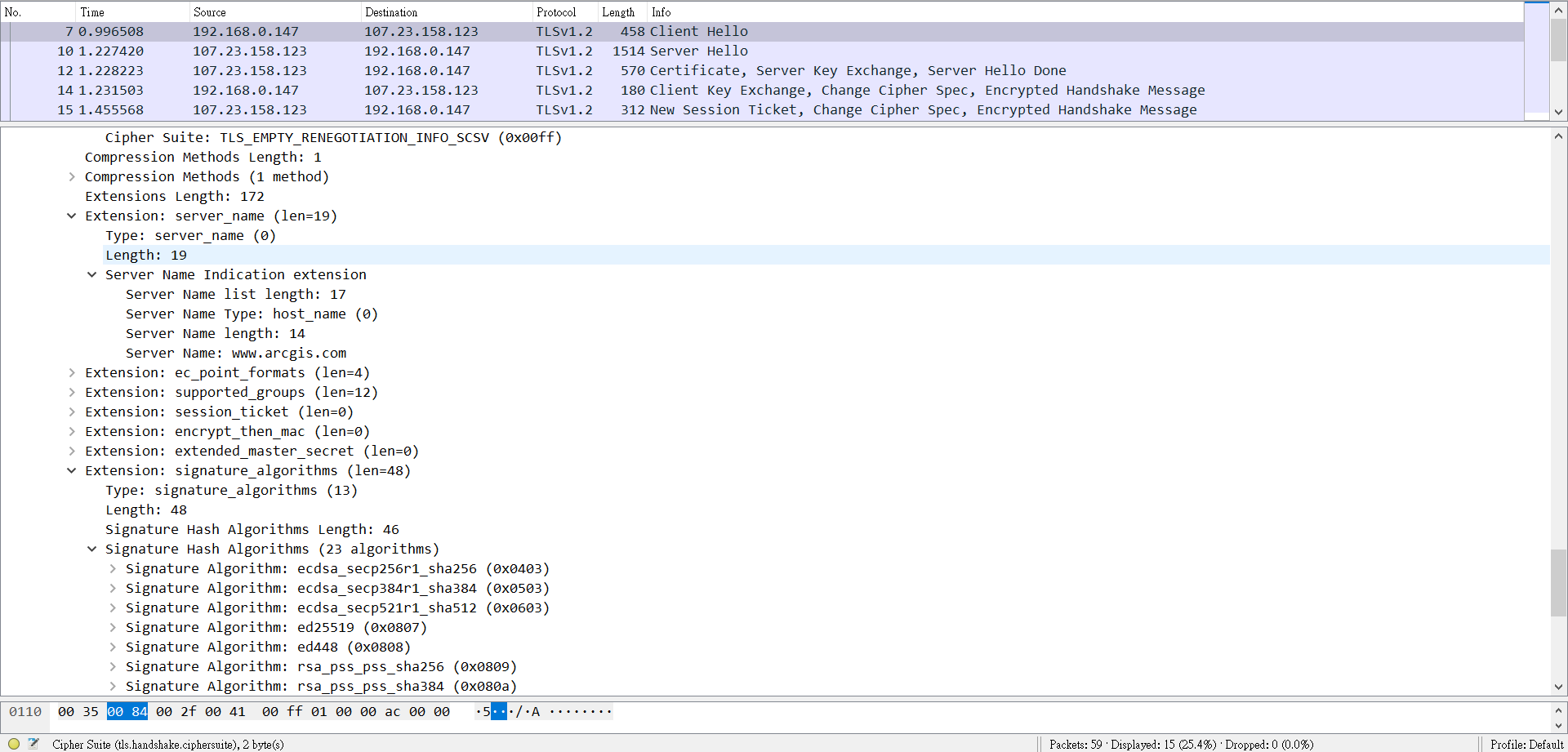
Time: Client’s local time, sometimes for browser fingerprinting

GMT Unix time: The current time and date in standard UNIX 32-bit format (seconds since the midnight starting Jan 1, 1970, UTC, ignoring leap seconds) according to the sender's internal clock. Clocks are not required to be set correctly by the basic TLS protocol

Random Bytes: 28 bytes generated by a secure random number generator.

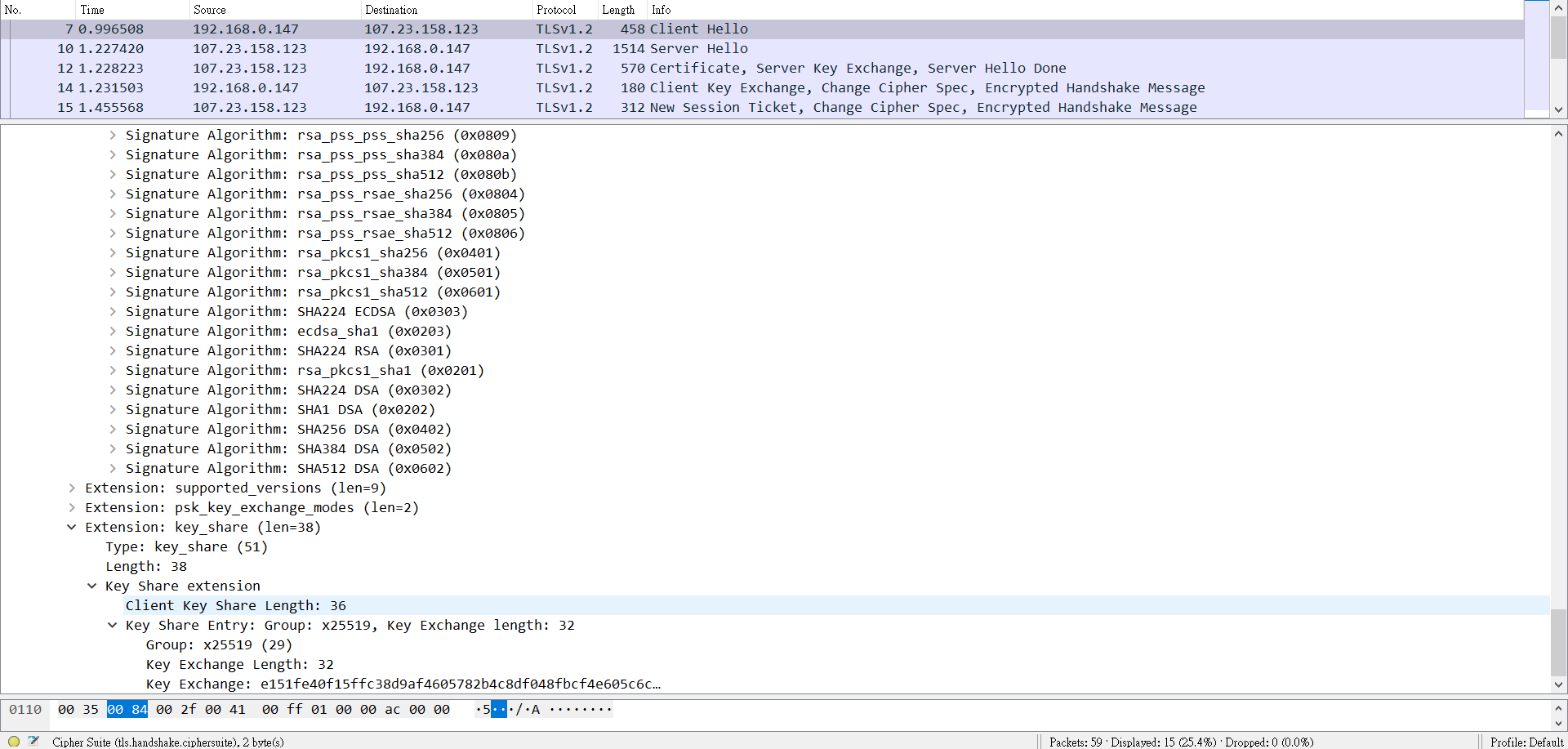
Session ID: The ID of a session the client wishes to use for this connection. A unique number used by the client to identify a session.

Cipher Suite: This is a list of the cryptographic options supported by the client, with the client's first preference first. The list of cipher suites supported by the client ordered by the client’s preference. The cipher suite consists of a key exchange algorithm, bulk encryption algorithm, MAC algorithm and a pseudorandom function. To sum up, it’s a list of all cipher suites supported by client in order of preference.



Compression Method: This is a list of the compression methods supported by the client, sorted by client preference. Contains a list of compression algorithms ordered by the client’s preference. This is optional.

Extension: Clients MAY request extended functionality from servers by sending data in the extensions field. The actual "Extension" format is defined in Section 7.4.1.4. Usually carry additional data

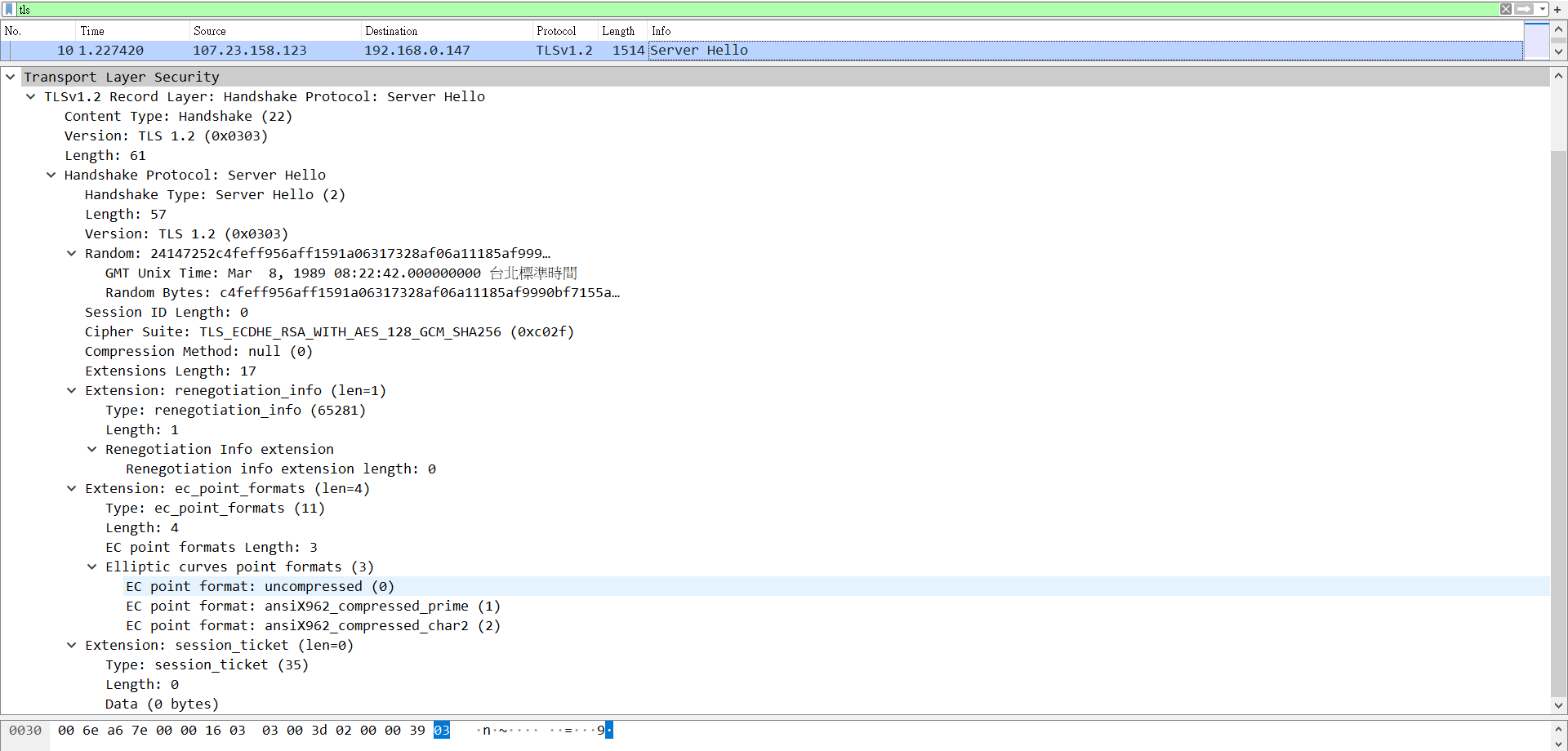


Extension: Clients MAY request extended functionality from servers by sending data in the extensions field. The actual "Extension" format is defined in Section 7.4.1.4. Usually carry additional data

1. Server response to Client：

No. 10 Server Hello：

The server will send this message in response to a ClientHello message when it was able to find an acceptable set of algorithms. If it cannot find such a match, it will respond with a handshake failure alert.



Record Layer Version: The TLS protocol version number that communication uses.

inner layer Version: This field will contain the lower of that suggested by the client in the client hello and the highest supported by the server. The highest TLS protocol version supported by the server which is also supported by the client.

Random: Generated by the server and MUST be independently generated from the ClientHello.random. Pseudorandom number used to generate the Master Secret.

Session ID: This is the identity of the session corresponding to this

connection. If the ClientHello.session\_id was non-empty, the server will look in its session cache for a match. If a match is found and the server is willing to establish the new connection using the specified session state, the server will respond with the same value as was supplied by the client. This indicates a resumed session and dictates that the parties must proceed directly to the Finished messages. Unique number to identify the session for the corresponding connection with the client.

If the session ID in the client hello message is not empty, the server will find a match in the session cache.

If a match is found and the server wants to use the same session state, it returns the same ID as sent by the client.

If the server doesn’t want to resume the same session, then a new ID is generated. The server can also send an empty ID, indicating the session cannot be resumed.

Cipher Suite: The single cipher suite selected by the server from the list in ClientHello.cipher\_suites. For resumed sessions, this field is the value from the state of the session being resumed. The single strongest cipher suite that both the server and the client support. If there is no supporting cipher suite, then a handshake failure alert is created.

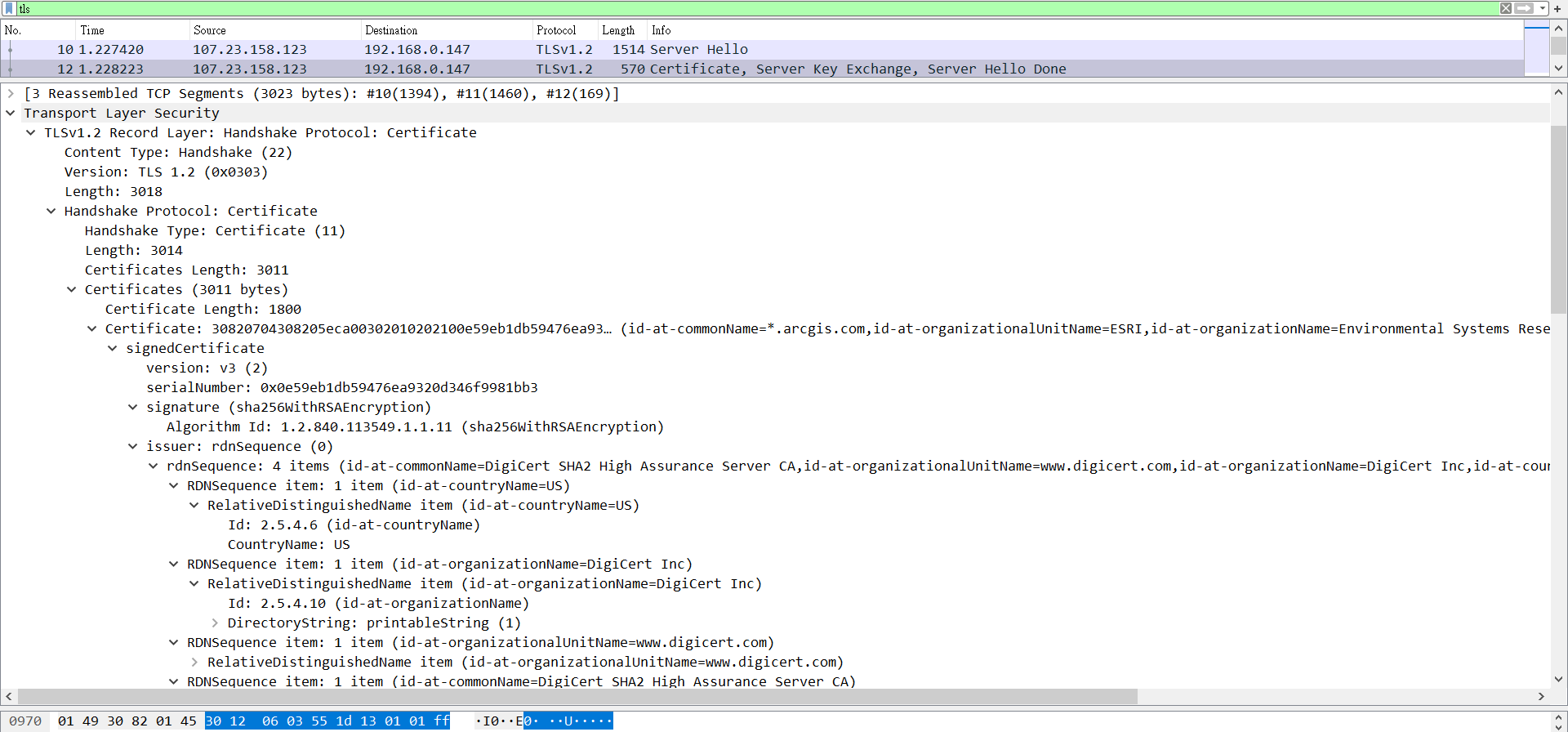
Compression Method: The single compression algorithm selected by the server from the list in ClientHello.compression\_methods. For resumed sessions, this field is the value from the resumed session state. The compression algorithm agreed by both the server and the client. This is optional.

No. 12 (Server) Certificate：

The server MUST send a Certificate message whenever the agreed-upon key exchange method uses certificates for authentication (this includes all key exchange methods defined in this document except DH\_anon). This message will always immediately follow the ServerHello message.

This message conveys the server's certificate chain to the client.

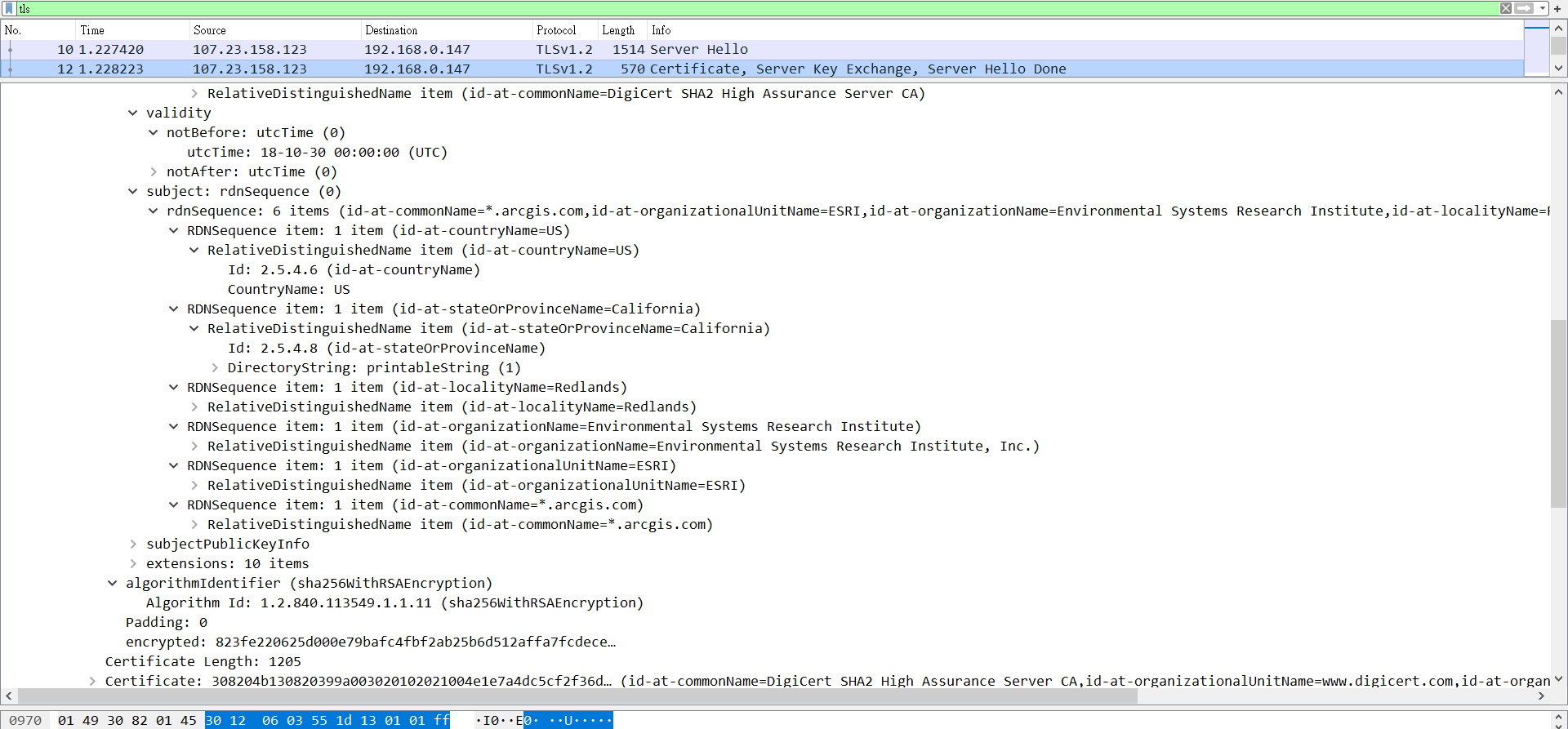
The certificate MUST be appropriate for the negotiated cipher suite's key exchange algorithm and any negotiated extensions.



Version: The TLS protocol version number that communication uses.

Algorithm Id: signature’s algorithm

Certificates: Certificate MUST come first in the list. Each following certificate MUST directly certify the one preceding it. Because certificate validation requires that root keys be distributed independently, the self-signed certificate that specifies the root certificate authority MAY be omitted from the chain, under the assumption that the remote end must already possess it in order to validate it in any case.



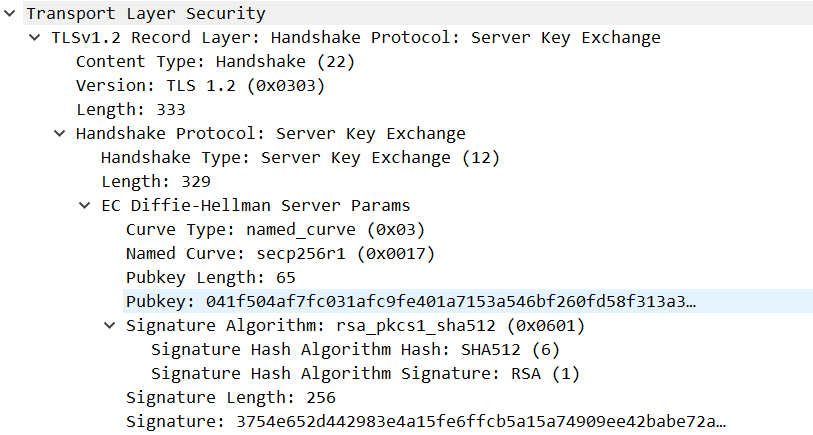
The server sends the client a list of X.509 certificates to authenticate itself. The server’s certificate contains its public key. This certificate authentication is either done by a third party (Certificate Authority) that is trusted by the peers, the operating system and the browser which contains the list of well-known Certificate Authorities or by manually importing certificates that the user trusts.

No. 12 Server Key Exchange：

This message will be sent immediately after the server Certificate message (or the ServerHello message, if this is an anonymous negotiation).

The ServerKeyExchange message is sent by the server only when the server Certificate message (if sent) does not contain enough data to allow the client to exchange a premaster secret.

This message conveys cryptographic information to allow the client to communicate the premaster secret: a Diffie-Hellman public key with which the client can complete a key exchange (with the result being the premaster secret) or a public key for some other algorithm.



Version: The TLS protocol version number that communication uses.

Pubkey: Public key

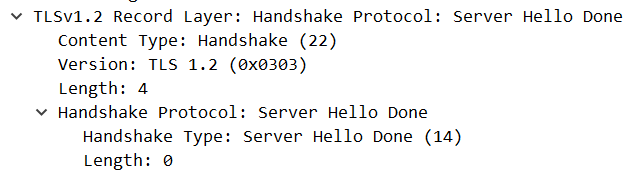
The message is optional and sent when the public key present in the server’s certificate is not suitable for key exchange or if the cipher suite places a restriction requiring a temporary key. This key is used by the client to encrypt Client Key Exchange later in the process.

No. 12 Server Hello Done：

The ServerHelloDone message is sent by the server to indicate the end of the ServerHello and associated messages. After sending this message, the server will wait for a client response.

This message means that the server is done sending messages to support the key exchange, and the client can proceed with its phase of the key exchange.

Upon receipt of the ServerHelloDone message, the client SHOULD verify that the server provided a valid certificate, if required, and check that the server hello parameters are acceptable



Version: The TLS protocol version number that communication uses.

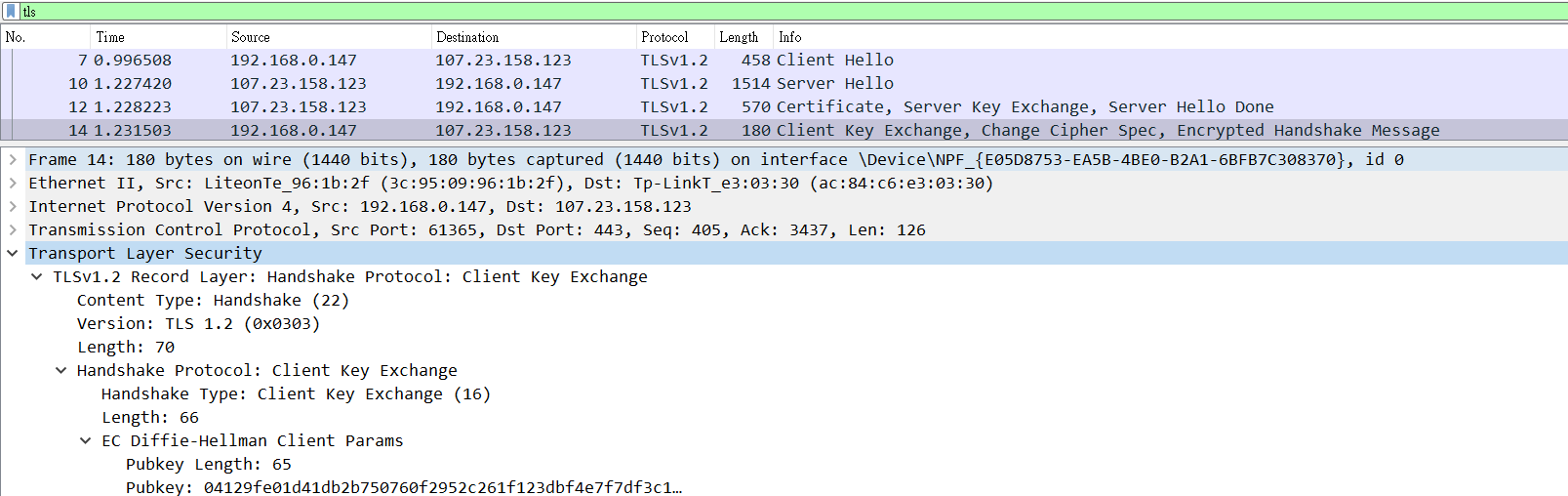
This message indicates the server is done and is awaiting the client’s response.

1. Client response to Server：

No. 14 Client Key Exchange：

This message is always sent by the client. It MUST immediately follow the client certificate message, if it is sent. Otherwise, it MUST be the first message sent by the client after it receives the ServerHelloDone message.

With this message, the premaster secret is set, either by direct transmission of the RSA-encrypted secret or by the transmission of Diffie-Hellman parameters that will allow each side to agree upon the same premaster secret.



Version: The TLS protocol version number that communication uses.

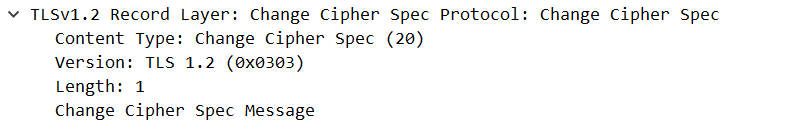
The protocol version of the client which the server verifies if it matches with the original client hello message.

Pubkey: public key

No. 14 Change Cipher Spec：

Signal transitions in ciphering strategies. The protocol consists of a single message, which is encrypted and compressed under the current connection state. The message consists of a single byte of value 1.

And the client copies the pending Cipher Spec into the current Cipher Spec.



Version: The TLS protocol version number that communication uses.

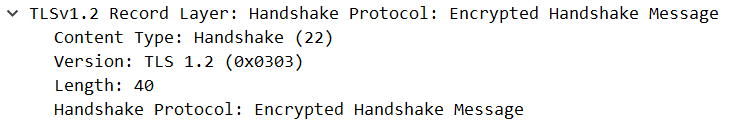
Length: 1Byte. Because the content is 1, which means to notify the server.

This message notifies the server that all the future messages will be encrypted using the algorithm and keys that were just negotiated.

No. 14 Encrypted Handshake Message(Finished)：

A Finished message is always sent immediately after a change cipher spec message to verify that the key exchange and authentication processes were successful. It is essential that a change cipher spec message be received between the other handshake messages and the Finished message.

The Finished message is the first one protected with the just negotiated algorithms, keys, and secrets. Recipients of Finished messages MUST verify that the contents are correct. Once a side has sent its Finished message and received and validated the Finished message from its peer, it may begin to send and receive application data over the connection.



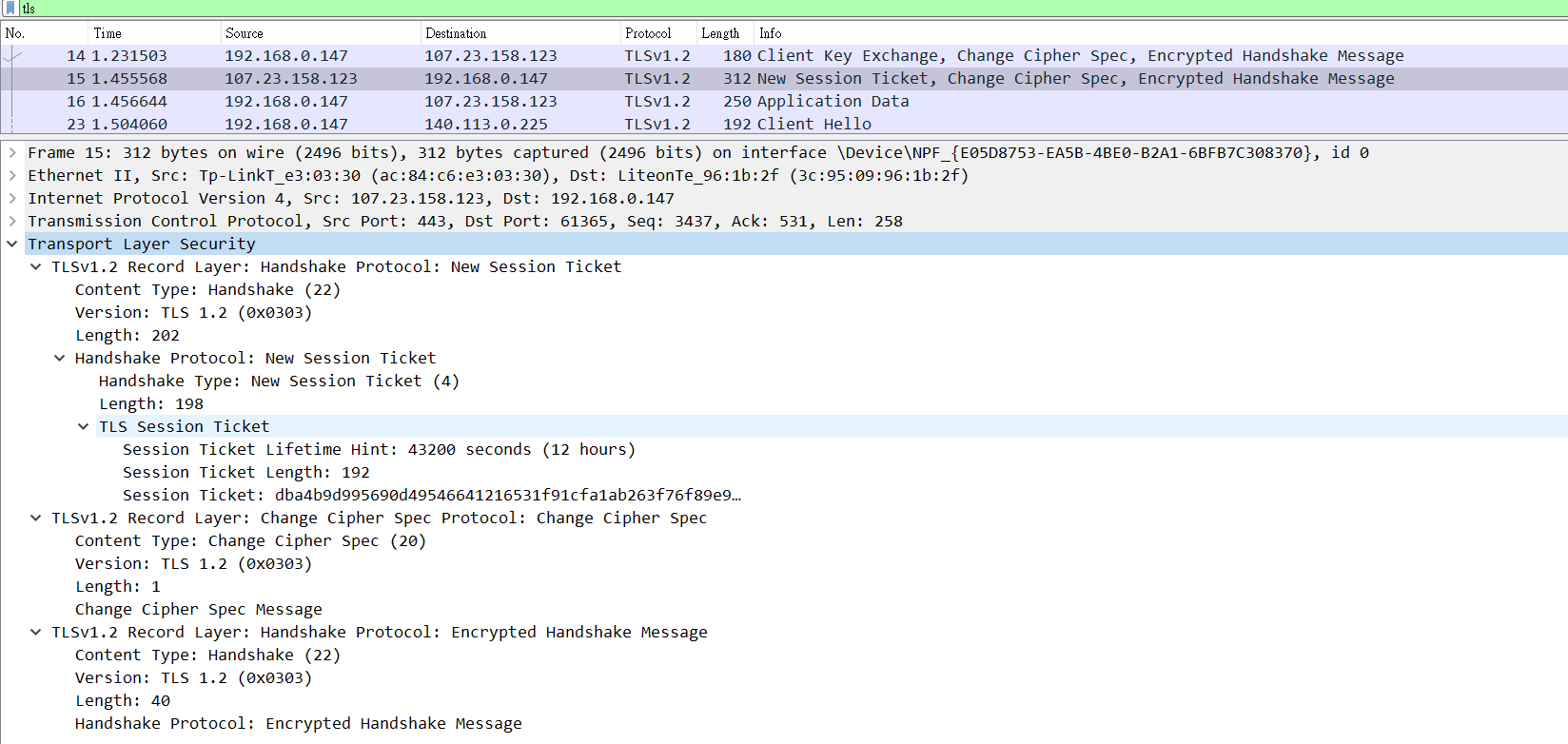
Version: The TLS protocol version number that communication uses.

1. Server response to Client：

No. 15 Change Cipher Spec：

Signal transitions in ciphering strategies. The protocol consists of a single message, which is encrypted and compressed under the current connection state. The message consists of a single byte of value 1.

Transfer the pending to the current Cipher Spec, and send its Finished message under the new Cipher Spec.



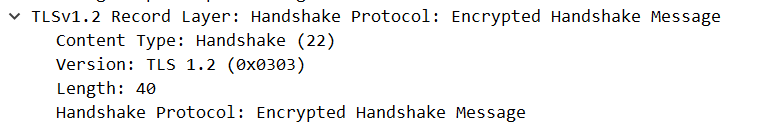
Version: The TLS protocol version number that communication uses.

Length: 1Byte. Because the content is 1, which means to inform the client.

The server informs the client that it the messages will be encrypted with the existing algorithms and keys. The record layer now changes its state to use the symmetric key encryption.

No. 15 Encrypted Handshake Massage(Finished)：

The Finished message is the first one protected with the just negotiated algorithms, keys, and secrets. Recipients of Finished messages MUST verify that the contents are correct. Once a side has sent its Finished message and received and validated the Finished message from its peer, it may begin to send and receive application data over the connection.



Version: The TLS protocol version number that communication uses.

Like the Client Finished message but contains a different label (“server finished”). Once the client successfully decrypts and validates the message, the server is successfully authenticated.